

LJD-SY-5100 学习实验开发板

用户手册

北京蓝海微芯科技发展有限公司

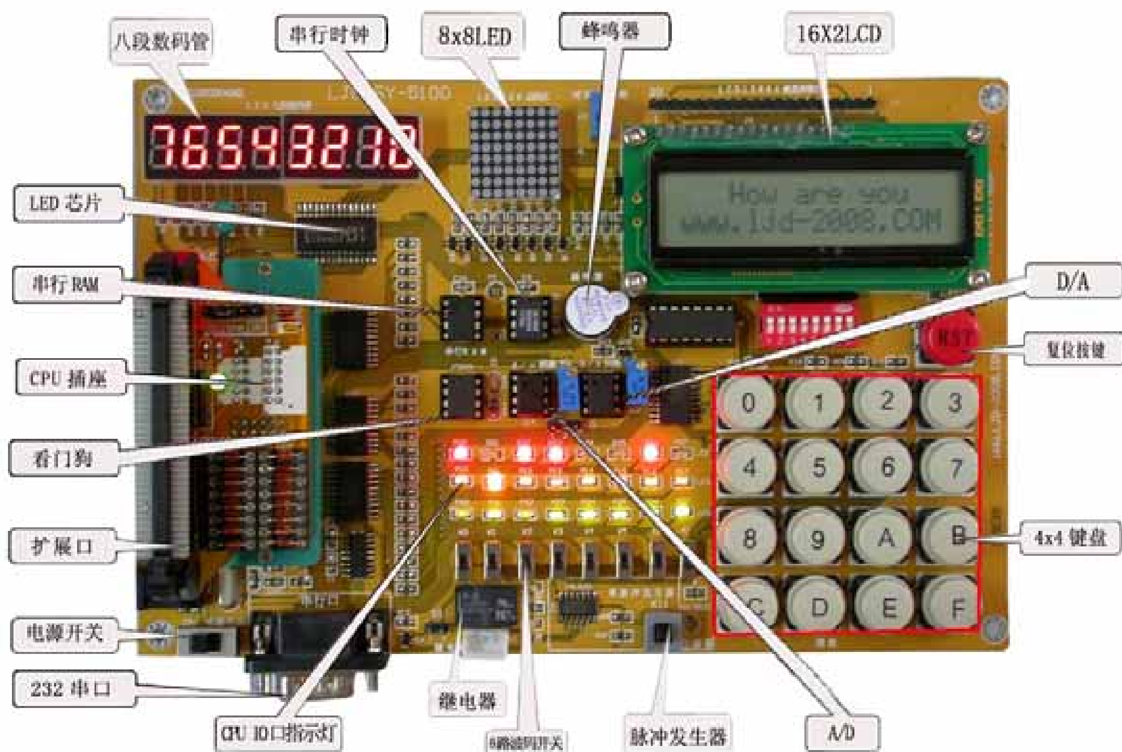
http://www.bluemcu.com

目　录

第一章 LJD-SY-5100 学习板介绍

1－1 实验板简介

LJD-SY-5100 实验板是一款非常适用的单片机学习和开发的工具，非常适合初步接触单片机的大中专学生学习使用，本实验板应用实例包括键盘扫描、数码管显示、字符型 LCD 显示、I^2C 接口 E^2PROM 24C02 及实时时钟 DS1302 的读写、串行 A/D、D/A 转换，继电器，RS232 串口通讯和蜂鸣器操作，可让使用者基本熟悉单片机及目前流行的芯片间的串行数据传输技术的设计与应用。提供实验电路的原理图和各个实验课题的程序，浅显易懂，入门方便。并提供调试监控，通过串口与 Keil C51 调试软件μVision2 配合，可实现单步、断点、连续等方式调试实际应用程序，省去购买仿真器。并提供可在系统编程的单片机 STC89C52RC，省去购买通用编程器，单片机在用户系统上即可下载/烧录用户程序。利用引出的 CPU 信号，可以连接自己的实际应用系统。LJD-SY-5100 实验板是集仿真调试、在线编程于一体的实验开发板。



1－2 性能特点

1．集仿真器，下载编程器，实验开发板，实用外设为一体。

2．采用黄金板电路设计，铝合金底壳，精美彩色包装盒。

3．精制的设计工艺，精美高性能插接件，进口按键、开关经久耐用，标准机器焊接工艺，学习板的功能，工业级的性能。

4．完整 200 多页的实验例程及教学课件，应用例程，实用例程。

5．丰富的硬件资源：24 个红、黄、绿 LED 指示灯、8 个 8 段 LED、1 个 8×8 矩阵 LED、2×16 的 LCD、4×4（键盘） 16 个进口按键（带键帽）、8 位拨动开关、单脉冲发生器、继电器、温度传感器、最新 I^2C 接口芯片，（RAM、看门狗、A/D、D/A、I/O）全部是实用控制中的最新技术，独一无二巧妙驱动分配电路，使单片机 I/O 的学习应用

系统发挥无极致又可连贯实验，也可外接用户的接口电路及扩展电路。

6．大量实用应用例程、专业的应用设计工程实践经验的积累。

7．编程及调试共用一个锁紧插座、下载和仿真采用同一个串行口连接更简单，CPU 为最新最流行的 STC89C52 工业级（兼容 AT89C51/52 等 CPU）。

8.配有 40PIN 外扩接口，可扩展其它外围电路，同时配有标准 40PIN 仿真头，可以把 SY-5100 当仿真器使用。

9．提供丰富的汇编 C51 源代码实验程序，并且提供更多应用程序，如：“万年历”、“密码锁”、“电压测试表”、“电脑时钟”、“交通灯控制”、“电子测温计”。

10．配详细结合原理图，分解原理图及芯片的性能介绍。

1—3 功能介绍

LJD-SY-5100 学习板自带调试监控，配合 Keil c 软件，在线下载、调试功能，目前支持下载的 CPU 为 STC89C51、89C52、89C51RC、89C54RD＋、89C58RD＋、89C516RD＋，配合其它编程器，可以支持其它更多种类的 CPU。

实验板实验资源和接口分配如下：

1．8 个 8 段 LED

2．1 个 8×8 矩阵 LED

3．24 个红、黄、绿高亮 LED

4．4×4 矩阵键盘（进口键带键帽）

5．8 位独立的拔码开关 （进口）

6．16×2 字符式 LCD

7．128×64 图形（汉字）液晶接口（选配件）。

8．TLC549　8 位 I^2C　A/D 转换。

9．TCC5615　10 位 I^2C　D/A 转换。

10．×25045　看门狗＋复位＋RAM

11．A24C02　I^2C EEPROM

12．PCF8572　I^2C 串口转并 I/O 口

13．DS1302　SPI 串行实时时钟

14．DS18B20　1-wine 温度传感器

15．CH451　SPI 键盘/LED 扫描电路

16．MAX232　专用串行口转换接口电路

17．一路继电器，由发光管代表吸合、断开

18．一个蜂鸣器

19．1 路单脉冲发生器可产生中断信号及计数信号

20．直流电机，步进电机（选配件）

21．并行口电路扩展口（8255、DS12887、62256 光电隔离、并口 A/D、D/A、打印机接口）（选配件）

1－4 仿真调试部分

LJD-SY-5100 可直接使用 Keil C51 集成开发环境，仿真功能扩大，同时还可通过附带的仿真电揽，可以直接对扩展口用户系统仿真。

仿真部分的功能特点如下：

1. 可以直接使用 Keil C51 集成环开发仿真。
2. 支持汇编、C51 源程序调试，支持混合在线调试。
3. 支持单步、断点、全部运行调试。
4. 可以观察数据区、变量区、工作区的数据变化。
5. 完整仿真 CPU P0、P1、P2、P3 口的全部特性。
6. 仿真频率可以达到 40MHZ 通讯波特率指 300bps-115200bps。
7. 可以仿真标准各种 51 内核的单片机仿真空间高达 63K。

1－5 在线编程部分

LJD-SY-5100 采用最新的 STC89 系列单片机作为用户终端 CPU，该 CPU 除具备一般 51 功能以外，同时可以具备工业级，高速度等优点，同时利用我们提供的下载调试可以直接通过串行口进行下载烧录，不需另外配套专用下载线。

1－6 标准配置

1. LJD-SY-5100 主机
2. RS232 串行口通讯，下载，烧录电缆

3. 5V 标准电源

4. 使用手册（转为 PDF 文档）

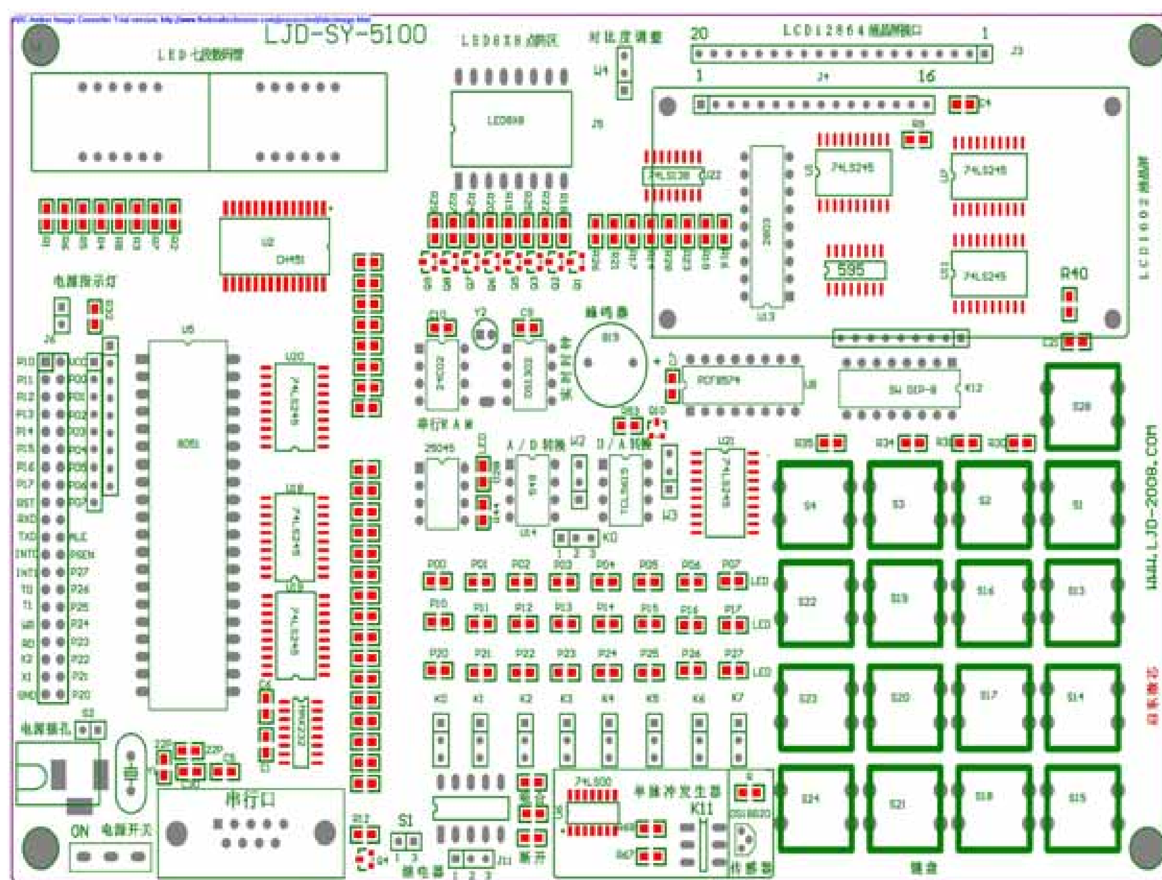
5. STC89C52　1 片

6. 调试监控　1 片

7. 扩展电缆　1 条

8. 光盘　　1 张

1—7 实验板结构图



1—8 实验板功能器件

名称	功能说明
七段数码管	8 个共阳 LED 数码管段位接 CH451 的 SEG 端，字位接 CH451 的 DIG 端
CH451	整合了数码管显示驱动和键盘扫描控制以及μP 监控的多功能外围芯片。
MCU	系统控制、编程、下载等
MAX232	RS232 串口通讯
继电器	在自动控制电路中起控制与隔离作用的执行部件，可以用低电压、小电流来控制大电流、高电压的自动开关。
25045	看门狗＋复位＋RAM
TLC549	8 位 I^2C A/D 转换
TLC5615	10 位 I^2C D/A 转换
24C02	I^2C EEPROM
DS1302	SPI 串行实时时钟
蜂鸣器	音乐程序演示，接 P3.7
PCF8574	I^2C 串口转并 I/0 口
DS18B20	一线温度传感器
LED8X8 点阵块	文字及图形显示实验
LCD1602	字符显示实验
红色	8 个红色发光二极管通过一片 74LS245 与 P0 口相连
黄色	8 个黄色发光二极管通过一片 74LS245 与 P1 口相连
绿色	8 个绿色发光二极管通过一片 74LS245 与 P2 口相连
矩阵键盘	4X4 矩阵键盘，行与列分别接 CH451 的 SEG 端（SEG0-SEG3）和 DIG 端（DIG0-DIG3）

1－9 应用接口和功能开关

标号	名称	功能说明
J6	仿真接口	通过连接 40PIN 仿真头对其他目标系统仿真调试,也可通过此接口引出 CPU 信号扩展其他实验
J3	LCD12864 液晶接口	此接口可用来外接一块 12864 汉字图形液晶屏
K0-K7	状态开关	8个状态开关分别与P2口的8位 IO 口相连，通过设置某位状态开关可将 P2 口的某一位设成低电平。
W4	对比度调整旋扭	此旋扭用来调整 LCD1602 液晶屏显示对比度

K12	拨码开关	将 2 拨通此时 8 位红色 LED 发光管与 P0 口接通，反之则断开。将 3 拨通此时 8 位绿色 LED 发光管与 P2 口接通反之则断开。将 4 拨通此时 8 位黄色 LED 发光管与 P1 口接通，反之则断开。将 5 拨通此时 PCF8574 的 8 位并口 IO 口与 8 位绿色 LED 发光管接通。同时拨通 7，8 此时 P0，P2 口与 LCD1602 接通这时 MCU 通过 P0，P2 口对液晶屏显示控制。拨通 6 此时 P2 口与 LED8X8 模块接通，这时 MCU 通过 P2 口对模块显示控制。
W2	TLC549 模拟电压输入旋扭	实验时通过旋转次旋扭改变输入电压值，A/D549 可以采集不同电压值进行转换。
W3	TCL5615 参考电压旋扭	旋转此旋扭，可以向 D/ATCL5615 提供合适的参考电压。

注：K12 拨码开关是用来断开和接通某个外设，具体每位作用见上表。在外设不互相影响的情况下用户可自由接通任何外设，可以接通一个，也可以同时接通多个。在这里用户可以自己发挥。例如在调试程序时就可以接通红黄绿发光管与 P0，P1，P2 口。用户通过观查各个发光管亮暗，就可知各个 IO 口数据状态（0 或 1）。

第二章 软件试验部分

2－1 CPU 片内清零程序

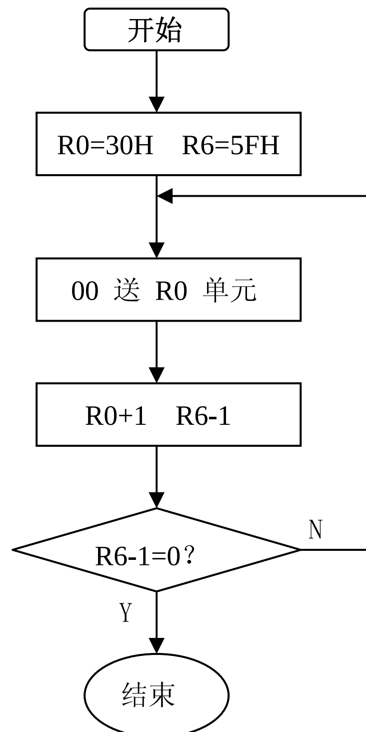
一、 实验目的

掌握 MCS-51 汇编语言的设计，了解单片机的寻址方式以及调试方法。

二、实验内容

把单片机片内的 30H～7FH 单元清零。

三、实验框图



四、实验步骤

用连续或者单步的方式运行程序，检查 30H-7FH 执行前后的内容变化。

五、实验思考

如果把 30H-7FH 的内容改为 55H，任何修改程序。

六、程序清单

文件名称：RAMCLR_1.ASM

```
       ORG 0000H
CLEAR:MOV  R0,#30H    ;30H 送 R0 寄存器
      MOV  R6,#4FH    ;4FH 送 R6 寄存器（计数）
CLR1: MOV  A,#00H      ;00 送累加器 A
```

```
        MOV   @R0,A          ;00 送到 30H-7FH 单元
        INC   R0             ;R0 加 1
        DJNZ   R6,CLR1       ;不到 4F 个字节再清
WAIT: LJMP    WAIT
        END
```

2－2 数据排序实验程序

一、 实验目的

掌握外部 RAM 中数据的操作方法以及调试方法

二、实验内容

用冒泡法将 RAM 中几个单字节无符号整数，按照从小到大的次序重新排列。

三、实验框图

见下图

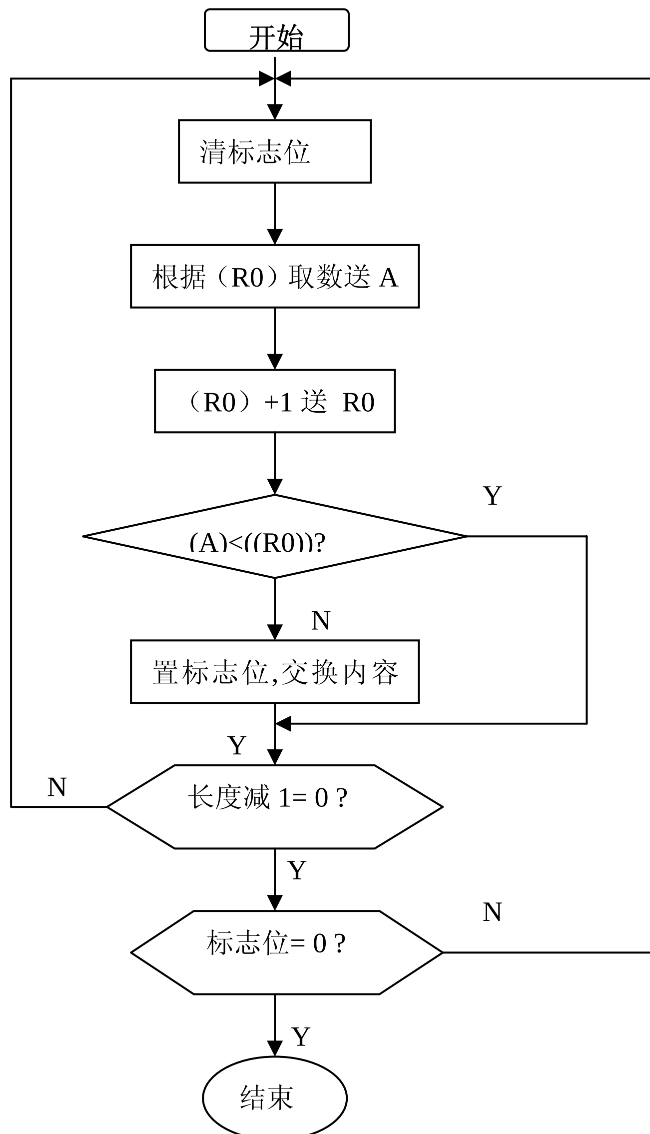
四、实验步骤

把 CPU 中 RAM 的 50H-5AH 中放入不等的数据，运行该程序，然后检查 50H-5AH 的数据是否按照从小到大排列。

五、思考

修改程序，把 50H-5AH 中的数据从大到小排列，任何设计程序

六、程序清单



;文件名称：SJPX.ASM

```
        ORG  0000H
PXCX:   MOV     R3,#50H
QL4:    MOV     A,R3        ;指针送 R0
        MOV     R0,A
        MOV     R7,#0AH        ;长度送 R7
        CLR     00H            ;标志位=0
```

```
        MOV       A,@R0
QL2:    INC     R0
        MOV     R2,A
        CLR     C
        MOV       22H,@R0
        CJNE    A,22H,QL3    ;相等吗?
        SETB    C
QL3:    MOV      A,R2
        JC      QL1          ;大于交换位置
        SETB    00H
        XCH      A,@R0
        DEC      R0
        XCH      A,@R0       ;大于交换位置
        INC      R0
QL1:    MOV      A,@R0
        DJNZ    R7,QL2
        JB      00H,QL4      ;一次循环中有继续交换
        SJMP    $          ;没有交换退出
        END
```

2－3 无符号双字节乘法实验程序

一、　实验目的

掌握 MCS-51 的计算指令的使用以及调试方法

二、实验内容

将（R2R3）和（R6R7）中双字节无符号整数相乘，积送 R4、R5、R6、R7。本程序是利用单字节乘法指令运算的。

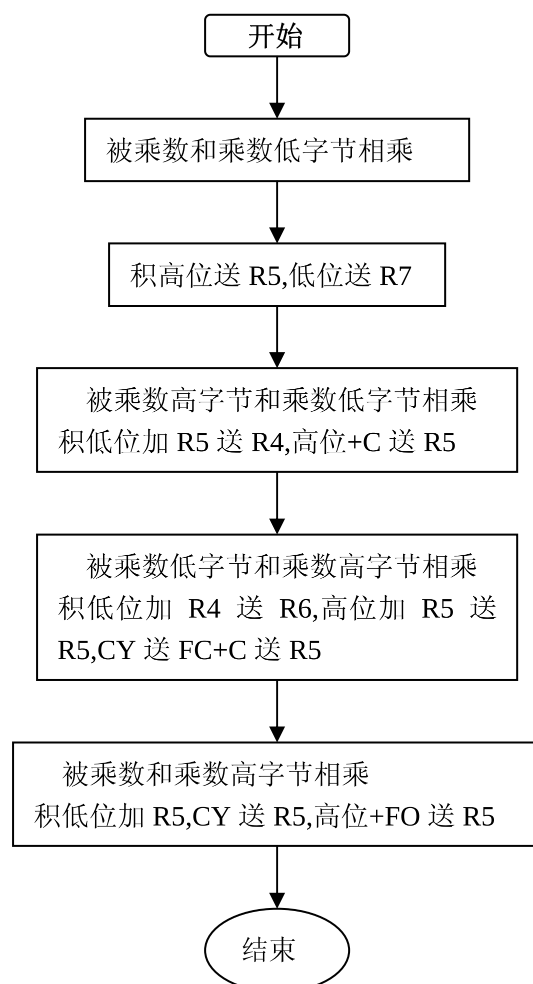
三、实验框图

见下图

四、实验步骤

在 R2R3 和 R6R7 中输入无符号整数，连续或者单步运行程序，然后检查 R4，R5，R6，R7 中的内容。

五、程序清单



;文件名称:WMUL.ASM

```
        ORG   0000H
QKUL:   MOV    A,R3
        MOV    B,R7
        MUL    AB          ;R3*R7
        XCH    A,R7        ;R7=(R3*R7)的低字节
        MOV    R5,B        ;R5=(R3*R7)的高字节
        MOV    B,R2
        MUL    AB          ；R2*R7
        ADD    A,R5
        MOV    R4,A
        CLR    A
        ADDC   A,B
        MOV    R5,A        ;R5=(R2*R7)的高字节
        MOV    A,R6
        MOV    B,R3
        MUL    AB          ;R3*R6
        ADD    A,R4
        XCH    A,R6
        XCH    A,B
        ADDC A,R5
```

```
        MOV     R5,A
        MOV     PSW.5,C         ;存 CY
        MOV     A,R2
        MUL     AB          ;R2*R6
        ADD     A,R5
        MOV     R5,A
        CLR     A
        MOV     ACC.0,C
        MOV     C,PSW.5         ;加上一次加法的进位.
        ADDC    A,B
        MOV     R4,A
        SJMP    $
        END
```

2－4 多分支转移实验程序

一、 实验目的

掌握单片机的汇编指令以及调试方法

二、实验内容

根据送入的数据转移到不同的地址。

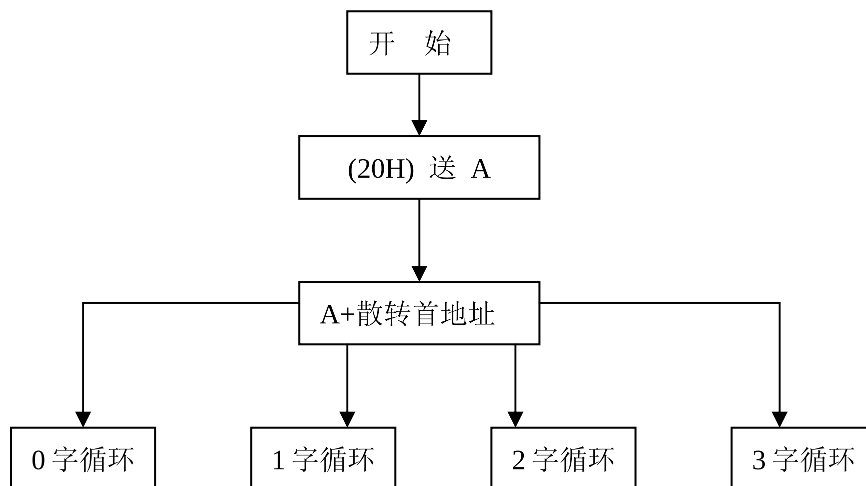
三、实验框图

见下图

四、实验步骤

在 20H 中送入 00H、01H、02H、03H 每次运行程序后，观察数码管上循环显示的数字的变化。

五、程序清单




```
;文件名称: DFZ.ASM
          TIMER0   EQU   30H
          TIMER1   EQU   31H
          ORG  0000H
    FZCX: MOV    A,20H
          MOV    DPTR,#TAB1    ;散转的首地址
          RL  A
          JMP    @A+DPTR
    TAB1: AJMP   FZ0         ; 0 字循环
          AJMP   FZ1         ; 1 字循环
          AJMP   FZ2         ; 2 字循环
```

```
        AJMP   FZ3        ; 3 字循环
  FZ0:  MOV    20H,#0C0H
        LJMP   XS
  FZ1:  MOV    20H,#0F9H
        LJMP   XS
  FZ2:  MOV    20H,#0A4H
        LJMP   XS
  FZ3:  MOV    20H,#0B0H
        LJMP   XS
  XS:      MOV    A,20H      ;显示子程序
        MOV    R0,#22H
        MOV    R1,#21H
        MOVX   @R0,A
        MOV    A,#01H
  XS1:  MOVX @R1,A
        MOV    R2,#20H
        LCALL DELAY1S
        RL  A
        SJMP   XS1
;延时子程序
DELAY:     PUSH   TIMER1;         延时 TIMER1*1 ms for 12MHz
        PUSH   TIMER0
```

```
DELAY1:     MOV    TIMER0,#250      ;循环一次需要 4 个机器周期，
时间为 4*1μS
DELAY2:     NOP           ;       1 个周期
            NOP           ;       1 个周期
            DJNZ   TIMER0,DELAY2   ;2 个周期
            DJNZ   TIMER1,DELAY1
            POP    TIMER0
            POP    TIMER1
            RET
DELAY1S:    NOP;              延时 1 sec
         PUSH   TIMER1
         MOV    TIMER1,#250
         LCALL DELAY
         LCALL DELAY
         LCALL DELAY
         LCALL DELAY
         POP    TIMER1
         RET
         NOP;---------------------
END
```

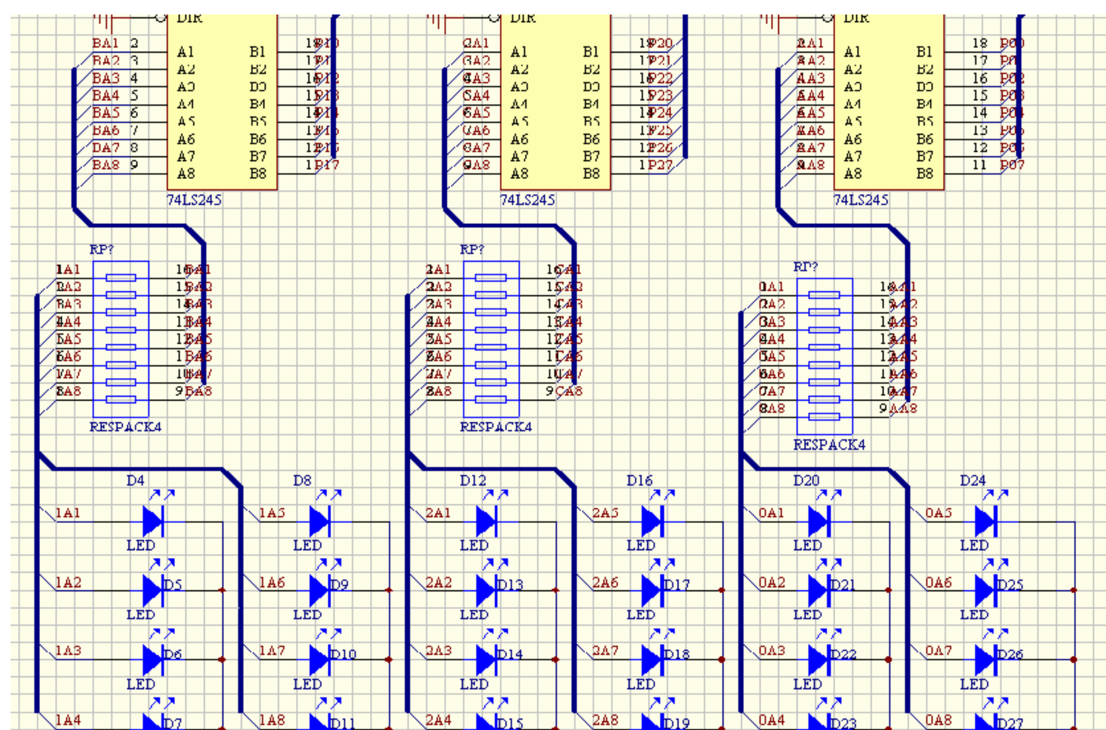
第三章 实验板电路分析与操作说明

3－1　I/O 口实验

<u>硬件设计与基本概念</u>

发光二极管 LED(Light Emitting Diode)为一种能够发光的半导体组件，常被用来当作电源指示灯或状态指示器。

LED 和二极管一样具有极性，当施以正向偏压时会发光，施以反向偏压时则不会发光；发光亮度与通过电流成正比。一般 LED 的工作电流在 15～20mA，若电流过大时会损坏 LED，必须串连一个限流电阻。



请注意：若要测试这个实验，请将拨码开关 K12 的 1，2，3，4 拨上去。

实验程序：红黄绿灯循环显示

```
        ORG  0030H
```

```
ST:     CLR P3.3
        MOV A,#0FEH
        MOV R1,#08H
        MOV R2,#08H
        MOV R3,#08H
ST0:    MOV P0,A
        RL A
        ACALL DELAY
        DJNZ R1,ST0
        MOV A,#0FEH
ST1:    MOV P1,A
        RL A
        ACALL DELAY
        DJNZ R2,ST1
        MOV A,#0FEH
ST2:    MOV P2,A
        RL A
        ACALL DELAY
        DJNZ R3,ST2
        SETB P3.3
SHT:    MOV A,#01H
        MOV R1,#08H
```

```
        MOV R2,#08H
        MOV R3,#08H
SHT0:   MOV P0,A
        RL A
        ACALL DELAY
        DJNZ R1,SHT0
        MOV A,#01H
SHT1:   MOV P1,A
        RL A
        ACALL DELAY
        DJNZ R2,SHT1
        MOV A,#01H
SHT2:   MOV P2,A
        RL A
        ACALL DELAY
        DJNZ R3,SHT2
        LJMP ST
DELAY:  MOV R6,#90H
H1 :    MOV R7,#0FFH
H2:     DJNZ R7,H2
        DJNZ R6,H1
        RET
```

END

3－2 键盘扫描与数码管显示

硬件设计与基本概念

１　ＣＨ４５１的功能与引脚介绍

ＣＨ４５１是一个整合了数码管显示驱动和键盘扫描控制以及μＰ监控的多功能外围芯片。ＣＨ４５１内置ＲＣ振荡电路，可以直接动态驱动８位数码管或者６４位ＬＥＤ，具有ＢＣＤ译码或不译码功能，可实现数据的左移、右移、左循环、右循环、各数字独立闪烁等控制功能。ＣＨ４５１内置大电流驱动级，段电流不小于３０ｍＡ，字电流不小于１６０ｍＡ，并有１６ 级亮度控制功能；在键盘控制方面，该器件内置６４键键盘控制器，可实现８×８矩阵键盘扫描，并内置去抖动电路，可提供按键中断与按键释放标志位等功能；在外部接口方面，ＣＨ４５１可选择简洁的１线串行接口或高速４线串行接口，且内置上电复位，可提供高电平有效复位和低电平有效复位两种输出，同时内置看门狗电路Ｗａｔｃｈ－Ｄｏｇ。ＣＨ４５１提供有２８引脚的ＤＩＰ２８与ＳＯＰ２８封装以及ＤＩＰ２４Ｓ封装形式，２８脚与２４脚在功能上稍有差别，它们的引脚定义见表１所列。

表 1 CH451 的引脚说明

28 脚引脚号	24 脚引脚号	引脚名称	类　型	引　脚　说　明
23	2	VCC	电源	正电源，持续电流不小于 200mA
9	15	GND	电源	接地，持续电流不小于 200mA
25	4	LOAD	输入	4 线串行接口的数据加协，带上拉
26	5	DIN	输入	4 线串行接口的数据输入，带上拉
27	6	DCLK	输入	串行接口的数据时钟，带上拉，可同时用于看门狗的清除输入
24	3	DOUT	输出	串行接口的数据输出键盘中断
22～15	1、24～18	SEG7～SEG0	三态输出及输入	数码管的段驱动，高电平有效，键盘扫描输入，高电平有效，带下拉
1～8	7～14	DIG7～DIG0	输出	数码管的字驱动，低电平有效，键盘扫描输入，高电平有效，带下拉
12	不支持	RST	输出	上电复位和看门狗复位，高电平有效
13	不支持	RST	输出	上电复位和看门狗复位，低电平有效
28	不支持	RSTI	输入	上电复位门限调整或手工复位输入
14	不支持	ADJ	输入	段电流上限调整，带强下拉
10	不支持	CLK	输入	外接阻容振荡
11	不支持	CLKO	输出	CLK 引脚时钟信号的二分频输出
	17	NC		不连接，禁止使用

2　ＣＨ４５１的操作命令

ＣＨ４５１的操作命令均为１２位，其中高４位为标识码，低８位为参数，各操作命令如下：

●空操作：００００ｘｘｘｘｘｘｘｘＢ（ｘ可为任意值，下同）

空操作命令对ＣＨ４５１不产生任何影响。该命令可以在多个ＣＨ４５１级联的应用中透过前级ＣＨ４５１向后级ＣＨ４５１发送操作命令而不影响前级ＣＨ４５１的状态。例如，要将操作命令００１０００００００１Ｂ发送给两级级联电路中的后级ＣＨ４５１（后级ＣＨ４５１的ＤＩＮ引脚连接到前级ＣＨ４５１的ＤＯＵＴ引脚），只要在该操作命令后添加空操作命令０００００００００００Ｂ再发送，那么，该操作命令将经过前级ＣＨ４５１到达后级ＣＨ４５１，而空操作命令留给了前级ＣＨ４５１。另外，为了在不影

响ＣＨ４５１的前提下变化ＤＣＬＫ以清除看门狗计时器，也可以发送空操作命令。在非级联的应用中，空操作命令可只发送高４位。

● 芯片内部复位：００１０００００００１Ｂ

内部复位命令可将ＣＨ４５１的各个寄存器和各种参数复位到默认的状态。芯片上电时，ＣＨ４５１均被复位，此时各个寄存器均复位为０，各种参数均恢复为默认值。

● 字数据移位：００１１００００００［Ｄ１］［Ｄ０］Ｂ

字数据移位命令共有４个：开环左移、右移，闭环左移、右移。Ｄ０为０时为开环，为１时为闭环，Ｄ１为０时左移，为１时为右移。开环左移时　ＤＩＧ０引脚对应的单元补００Ｈ，此时不译码方式显示为空格，ＢＣＤ译码方式时显示为０；开环右移时，ＤＩＧ７引脚对应的单元补００Ｈ；而在闭环时　ＤＩＧ０与ＤＩＧ７头尾相接，闭环移位。

● 设定系统参数：０１０００００００［ＷＤＯＧ］［ＫＥＹＢ］［ＤＩＳＰ］Ｂ

该命令用于设定ＣＨ４５１的系统级参数　如看门狗使能ＷＤＯＧ、键盘扫描使能ＫＥＹＢ、显示驱动使能ＤＩＳＰ等。各个参数均可通过１位数据来进行控制，将相应的数据位置为１可启用该功能，否则关闭该功能（默认值）。

● 设定显示参数：０１０１[ＭＯＤＥ][ＬＩＭＩＴ][ＩＮＴＥＮＳＩＴＹ]Ｂ

此命令用于设定ＣＨ４５１的显示参数,如译码方式ＭＯＤＥ（１位）、扫描极限ＬＩＭＩＴ（３位）、显示亮度ＩＮＴＥＮＳＩＴＹ（４位）等。译码方式ＭＯＤＥ为１时选择ＢＣＤ译码方式，为０时选择不译码方式。ＣＨ４５１默认工作于不译码方式，此时８个数据寄存器中字节数据的位７～位０分别对应８个数码管的小数点和段Ｇ～段Ａ，当数据位为１时，对应的数据段（或发光管）点亮；数据位为０时熄灭。ＣＨ４５１工作于ＢＣＤ译码方式主要应用于数码管驱动，单片机只要给出二进制数的ＢＣＤ码，便可由ＣＨ４５１将其译码并直接驱动数码管以显示对应的字符。ＢＣＤ译码方式是对数据寄存器中字节数据的位４～位０进行兼容ＢＣＤ的译码，可用于控制段驱动引脚ＳＥＧ６～ＳＥＧ０的输出，它们对应于数码管的段Ｇ～段Ａ，同时可用字节数据的位７控制段来驱动引脚ＳＥＧ７的输出以对应数码管的小数点，字节数据的位６和位５不影响ＢＣＤ译码的输出，它们可以是任意值。将位４～位０进行ＢＣＤ译码可显示以下２８个字符，其中０００００Ｂ～０１１１１Ｂ分别对应于“ ０～Ｆ ”、１００００Ｂ～１１０１０Ｂ分别对应于“ ” 空格 、“ ＋ ” ＋或加号 、“ － ” 负号或减号 、“ ＝ ” 等于号 、“ " 左方括号 、“ ” 右方括号 、“ ＿ ” 下划线 、“ Ｈ ”、“ Ｌ ”、“ Ｐ ”、“ ． ” 小数点 、其余值为空格。

扫描极限ＬＩＭＩＴ控制位００１Ｂ～１１１Ｂ和０００Ｂ（默认值）可分别设定扫描极限１～７和８。显示亮度ＩＮＴＥＮＳＩＴＹ控制位的０００１Ｂ～１１１１Ｂ和００００Ｂ（默认值）则用于分别设定显示驱动占空比１／１６～１５／１６ 和１６／１６，以实现１６级显示亮度控制。

●设定闪烁控制：０１１０[Ｄ７Ｓ][Ｄ６Ｓ][Ｄ５Ｓ][Ｄ４Ｓ][Ｄ３Ｓ][Ｄ２Ｓ][Ｄ１Ｓ][Ｄ０Ｓ]Ｂ

设定闪烁控制命令用于设定ＣＨ４５１的闪烁显示属性，其中Ｄ７Ｓ～Ｄ０Ｓ 分别对应于８个字驱动ＤＩＧ７～ＤＩＧ０。闪烁属性Ｄ７Ｓ～Ｄ０Ｓ分别通过１位数据控制，将相应的数据位置为１可使能闪烁显示，否则为正常显示，不闪烁（默认值）。

●加载字数据：１[ＤＩＧ＿ＡＤＤＲ][ＤＩＧ＿ＤＡＴＡ]Ｂ

加载字数据命令用于将字节数据ＤＩＧ＿ＤＡＴＡ（８位）写入ＤＩＧ＿ＡＤＤＲ（３位）指定的数据寄存器中。ＤＩＧ＿ＡＤＤＲ的０００Ｂ～１１１Ｂ分别用于指定数据寄存器的地址０～７，并分别对应于ＤＩＧ０～ＤＩＧ７引脚驱动的８个数码管。ＤＩＧ＿ＤＡＴＡ为待写入的字节数据。

●读取按键代码：０１１１ｘｘｘｘｘｘｘｘＢ

读取按键代码命令用于获得ＣＨ４５１最近检测到的有效按键的按键代码。该命令是唯一的具有数据返回的命令，ＣＨ４５１通常从ＤＯＵＴ引脚输出按键代码，按键代码总是７位数据，最高位是状态码，位５～位０是扫描码。读取按键代码命令的位数据Ｂ７～Ｂ０可以是任意值，所以控制器可以将该操作命令缩短为４位数据Ｂ１１～Ｂ８。例如，ＣＨ４５１检测到有效按键并中断时，如按键代码是５ＥＨ，则先向ＣＨ４５１发出读取按键代码命令０１１１Ｂ，然后再从ＤＯＵＴ获得按键代码５ＥＨ。

ＣＨ４５１所提供的按键代码为７位，位２～位０是列扫描码，位５～位３是行扫描码，位６是状态码（键按下为１，键释放为０）。例如，连接ＤＩＧ３与ＳＥＧ４的键被按下时，按键代码为６３Ｈ，键被释放后，按键代码是２３Ｈ。单片机可以在任何时候读取按键代码，但一般在ＣＨ４５１检测到有效按键而产生键盘中断时读取按键代码，此时按键代码的位６总是１。另外，如果需要了解按键何时释放，单片机可以通过查询方式定期读取按键代码，直到按键代码的位６为０。表２是连接在ＤＩＧ７～ＤＩＧ０与ＳＥＧ７～ＳＥＧ０之间的键被按下时，ＣＨ４５１所提供的按键代码。这些按键代码具有一定的规律，如果需要键被释放时的按键代码，可将表２中的按键代码的位６ 置０，也可将表中的按键代码减去４０Ｈ。应注意的是：ＣＨ４５１不支持组合键，也就是说，同一时刻，不能有两个或者更多的键被按下。

表 2 CH451 的键盘编码表

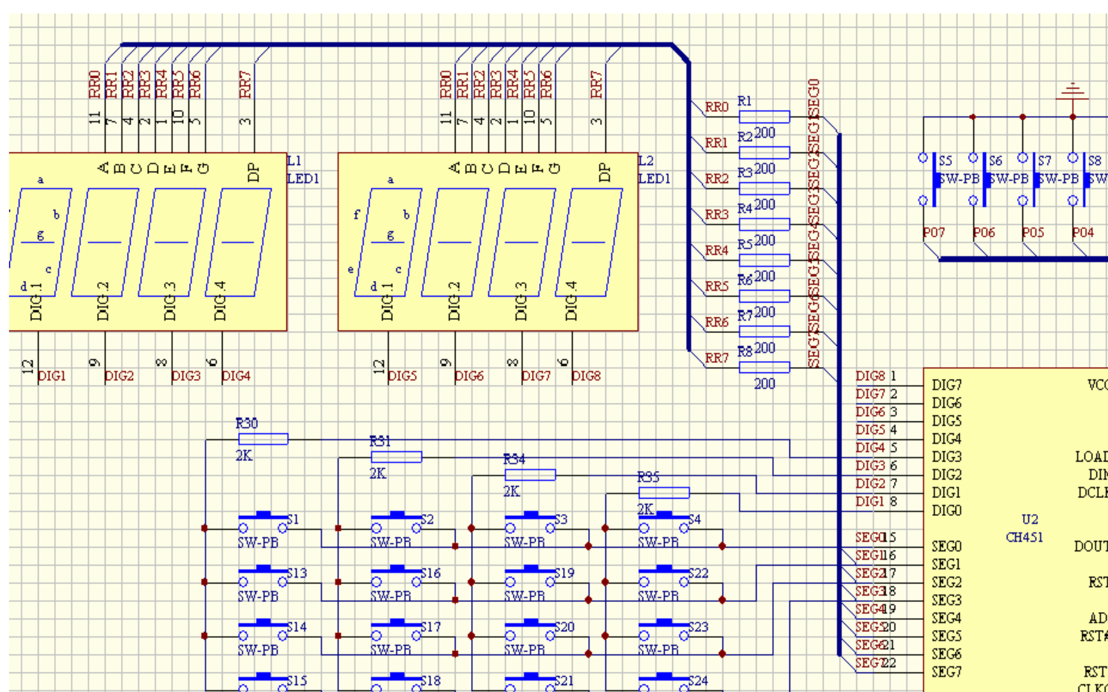
按键代码	DIG7	DIG6	DIG5	DIG4	DIG3	DIG2	DIG1	DIG0
SEG0	47H	46H	45H	44H	43H	42H	41H	40H
SEG1	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H
SEG2	57H	56H	55H	54H	53H	52H	51H	50H
SEG3	5FH	5EH	5DH	5CH	5BH	5AH	59H	58H
SEG4	67H	66H	65H	64H	63H	62H	61H	60H
SEG5	6FH	6EH	6DH	6CH	6BH	6AH	69H	68H
SEG6	77H	76H	75H	74H	73H	72H	71H	70H
SEG7	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H

3　串行接口应用电路

ＣＨ４５１与ＭＣＳ－５１单片机的连接参考原理图，其中ＤＯＵＴ引脚最好连接到单片机的中断输入引脚，这样可用中断方式响应按键。如果连接到非中断输入引脚，则应该使用查询方式确定ＣＨ４５１是否检测到有效按键，同时还可向单片机提供复位信号ＲＥＳＥＴ并带Ｗａｔｃｈ－Ｄｏｇ功能。ＣＨ４５１的段驱动引脚串接的电阻Ｒ１（２００Ω）用于限制和均衡段驱动电流。在５Ｖ电源电压下，串接２００Ω电阻通常对应１３ｍＡ段电流。ＣＨ４５１具有６４键的键盘扫描功能，为了防止键被按下后在ＳＥＧ信号线与ＤＩＧ信号线之间形成短路而影响数码管显示，一般应在ＣＨ４５１的ＤＩＧ０～ＤＩＧ７引脚与键盘矩阵之间串接限流电阻Ｒ２，其阻值可以从１ｋΩ至１０ｋΩ。

将Ｐ1.6与ＤＩＮ连接可用于输入串行数据，串行数据输入的顺序是低位在前，高位在后。另外，在上电复位后，ＣＨ４５１ 默认选择１线串行接口，如需选择４线串行接口，则应在ＤＣＬＫ输出串

行时钟之前，先在ＤＩＮ上输出一个低电平脉冲，以通知ＣＨ４５１为４线串行接口。将Ｐ1.7与ＤＣＬＫ连接可提供串行时钟，以使ＣＨ４５１在其上升沿从ＤＩＮ输入数据，并在其下降沿从ＤＯＵＴ输出数据。ＬＯＡＤ用于加载串行数据，ＣＨ４５１一般在其上升沿加载移位寄存器中的１２位数据以作为操作命令进行分析并处理。也就是说，ＬＯＡＤ的上升沿是串行数据帧的帧完成标志，此时无论移位寄存器中的１２位数据是否有效，ＣＨ４５１都会将其当作操作命令来处理。应注意的是，在级联电路中，单片机每次输出的串行数据必须是单个ＣＨ４５１的串行数据的位数乘以级联的级数。



实例程序：

```
;***********************************
;CH451 测试程序查表轮显 0--F
; LJD-SY-5100\example\ch4511\ch451.ASM
```

```
;**********************************
LOAD        BIT      P1.2
DIN         BIT      P1.0
DCLK        BIT      P1.1
DOUT        BIT      P3.3
;**********************************
DATA_F      DATA     034H
DATA_KEY    DATA     035H
TIMER       DATA     030H
TIMER1      DATA     031H
TIMER2      DATA     032H
TIMER3      DATA     033H
;***********************************
            ORG      0000H
       JMP      START
            ORG      0013H
            LJMP     CH451_INT1
       ORG      080H

START:

             MOV      SP,#60H
```

```
;*************************************
        MOV P1,#60H                 ;禁止其它芯片
         CLR      DIN                ;初始化 CH451
    SETB     DCLK
    SETB     DIN
    SETB     LOAD
    SETB     DOUT
         NOP
    MOV      B,#04H           ;设置 CH451
    MOV      A,#03H           ;关看门狗开显示键盘
    nop
    LCALL    WRITE
    NOP
START1:
    CLR      IT1                ;置外部信号为低电平触发
    CLR      IE1                 ;清中断标志
    SETB     PX1
    SETB     EX1                 ;允许键盘中断
    SETB     EA                 ;开总中断
    MOV       R5,#00H
TT1:
```

```
        MOV     A,R5
        LCALL   TT
        MOV     B,#08H          ;加载字数据 1
        LCALL   WRITE
        LCALL   DELAY_1S
        LCALL   DELAY_1S
        MOV     B,#03H          ;字数据左移
        MOV     A,#00H
        LCALL   WRITE
        INC     R5
        CJNE    R5,#010H,TT1
        JMP     START1
TT:
        MOV     DPTR,#TAB
        MOVC    A,@A+DPTR
        RET
        NOP
        JMP     START1
TAB:
        DB      03FH            ;0
        DB      006H            ;1
        DB      05BH            ;2
```

```
        DB      04FH            ;3
        DB      066H            ;4
        DB      06DH            ;5
        DB      07DH            ;6
        DB      07H             ;7
        DB      07FH            ;8
        DB      06FH            ;9
        DB      77H             ;A
        DB      07CH            ;B
        DB      039H            ;C
        DB      5EH             ;D
        DB      079H            ;E
        DB      071H            ;F
;***************键盘处理*****************************
CH451_INT1:
        PUSH    PSW               ;现场保护
        PUSH    ACC
        PUSH    B
        MOV     R4,#06H
YY:
        MOV     B,#08H
        MOV     A,#00H
```

```
        LCALL   WRITE
        MOV     B,#03H          ;字数据左移
        MOV     A,#00H
        LCALL   WRITE
        DJNZ    R4,YY
        LCALL   INTER
K1:
        MOV     R3,DATA_KEY
        CJNE    R3,#40H,K2
        JMP     LED_0
K2:
        MOV     R3,DATA_KEY
        CJNE    R3,#41H,K3
        JMP     LED_1
K3:
        MOV     R3,DATA_KEY
        CJNE    R3,#42H,K4
        JMP     LED_2
K4:
        MOV     R3,DATA_KEY
        CJNE    R3,#43H,K5
        JMP     LED_3
```

```
K5:
        MOV     R3,DATA_KEY
        CJNE    R3,#48H,K6
        JMP     LED_4
K6:
        MOV     R3,DATA_KEY
        CJNE    R3,#49H,K7
        JMP     LED_5
K7:
        MOV     R3,DATA_KEY
        CJNE    R3,#4AH,K8
        JMP     LED_6
K8:
        MOV     R3,DATA_KEY
        CJNE    R3,#4BH,K9
        JMP     LED_7
K9:
        MOV     R3,DATA_KEY
        CJNE    R3,#50H,K10
        JMP     LED_8
K10:
        MOV     R3,DATA_KEY
```

```
        CJNE    R3,#51H,K11
        JMP     LED_9
K11:
        MOV     R3,DATA_KEY
        CJNE    R3,#52H,K12
        JMP     LED_A
K12:
        MOV     R3,DATA_KEY
        CJNE    R3,#53H,K13
        JMP     LED_B
K13:
        MOV     R3,DATA_KEY
        CJNE    R3,#58H,K14
        JMP     LED_C
K14:
        MOV     R3,DATA_KEY
        CJNE    R3,#59H,K15
        JMP     LED_D
K15:
        MOV     R3,DATA_KEY
        CJNE    R3,#5AH,K16
        JMP     LED_E
```

```
K16:
        MOV      R3,DATA_KEY
        CJNE     R3,#5BH,K17
        JMP      LED_F
K17:
        POP   ACC
        POP   PSW
        CLR   IE1
        RETI
        NOP
        LJMP      START
LED_A:
         MOV      B,#08H
         MOV      A,#077H
         LCALL   WRITE
         JMP      DELAY1
         NOP
        LJMP      START
LED_B:
           MOV      B,#08H
           MOV      A,#07CH
           LCALL   WRITE
```

```
        JMP     DELAY1
        NOP
        LJMP     START
LED_C:
         MOV     B,#08H
         MOV     A,#039H
         LCALL   WRITE
         JMP     DELAY1
         NOP
         LJMP     START
LED_D:
         MOV     B,#08H
         MOV     A,#05EH
         LCALL   WRITE
         JMP     DELAY1
         NOP
         LJMP     START
LED_E:
        MOV     B,#08H
        MOV     A,#079H
        LCALL   WRITE
        JMP     DELAY1
```

```
        NOP
        LJMP     START
LED_F:
        MOV     B,#08H
        MOV     A,#071H
        LCALL   WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_0:
        MOV     B,#08H
        MOV     A,#03FH
        LCALL   WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_1:
        MOV     B,#08H
        MOV     A,#06H
        LCALL   WRITE
        JMP     DELAY1
        NOP
```

```
        LJMP     START
LED_2:
        MOV      B,#08H
        MOV      A,#05BH
        LCALL    WRITE
        JMP      DELAY1
        NOP
        LJMP     START
LED_3:
        MOV      B,#08H
        MOV      A,#04FH
        LCALL    WRITE
        JMP      DELAY1
        NOP
        LJMP     START
LED_4:
        MOV      B,#08H
        MOV      A,#066H
        LCALL    WRITE
        JMP      DELAY1
        NOP
        LJMP     START
```

```
LED_5:
        MOV     B,#08H
        MOV     A,#06DH
        LCALL   WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_6:
        MOV     B,#08H
        MOV     A,#07DH
        LCALL   WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_7:
        MOV     B,#08H
        MOV     A,#007H
        LCALL   WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_8:
```

```
        MOV     B,#08H
        MOV     A,#07FH
        LCALL   WRITE
        JMP     DELAY1
        NOP
        LJMP     START
LED_9:
        MOV     B,#08H
        MOV     A,#06FH
        LCALL   WRITE
        JMP     DELAY1
         NOP
        LJMP     START
DELAY1:
        CLR     IT1                     ;置外部信号为低电平触发
        CLR     IE1                     ;清中断标志
        SETB    PX1
        SETB    EX1                     ;允许键盘中断
        SETB    EA
        LCALL    DELAY_1S
       ;LCALL    DELAY_1S
       ;LCALL    DELAY_1S
```

```
        ;LCALL   DELAY_1S
         POP      B
         POP      ACC
         POP      PSW
         RETI
         NOP
         LJMP     START
;*********************************
WRITE:
         PUSH     ACC
         CLR       EX0
         CLR       LOAD
         MOV       R7,#08H
WRITE_1:
         RRC       A
         CLR       DCLK
         MOV       DIN,C
         SETB      DCLK
         DJNZ      R7,WRITE_1
         MOV       A,B
         MOV       R7,#004H
WRITE_2:
```

```
        RRC         A
        CLR         DCLK
        MOV         DIN,C
        SETB        DCLK
        DJNZ        R7,WRITE_2
        SETB        LOAD
        SETB        EX1
        POP         ACC
        RET
;*********************************
INTER:
        PUSH   PSW                  ;现场保护
        PUSH   ACC
        CLR         EX1
        CLR    LOAD                 ;命令开始
        MOV    A,#0F7H              ;读键值命令,忽略 12 位命令的低
8 位,高 4 位用作结束标志
INTER_4:
        SETB   C                    ;在高位添 0 以检测位数据结束
        RRC    A                    ;低位在前,高位在后
        CLR    DCLK
        MOV    DIN,C                   ;送出一位数据
```

```
        SETB   DCLK                    ;产生时钟上升沿锁通知CH451
输入位数据
        CJNE   A,#0FFH,INTER_4         ;位数据未完继续,共 4 位,完
成后为 0FFH
        SETB   LOAD                    ;产生加载上升沿通知 CH451
处理命令数据
        MOV    A,#0FCH                 ;该数据用以检测位数据结束
INTER_7:
        MOV    C,DOUT                  ;读入一位数据
        CLR    DCLK                    ;产生时钟下升沿通知 CH451 输
出下一位
        RLC    A                       ;数据移入ACC,高位在前,低位在后
    SETB   DCLK
    JC     INTER_7                     ;位数据未完继续,共 7 位,完
成后才移出 0
    MOV    DATA_KEY,A                  ;保存键值
    INC    DATA_F
    POP    ACC
    POP    PSW
    SETB   EX1
    CLR    IE1                         ;清中断标志,该指令需根据实际情况
作修改
```

```
    RET
;**********************************
DELAY_1S:
                MOV      TIMER1,#1
TEST_DYA:       MOV      TIMER2,#255
TEST_DYA1:      MOV      TIMER3,#255
TEST_DYA2:      NOP
        NOP
        DJNZ     TIMER3,TEST_DYA2
        DJNZ     TIMER2,TEST_DYA1
        DJNZ     TIMER1,TEST_DYA
        RET
END
```

3－3 字符型 LCD 显示

硬件设计与基本概念

1. 引脚功能

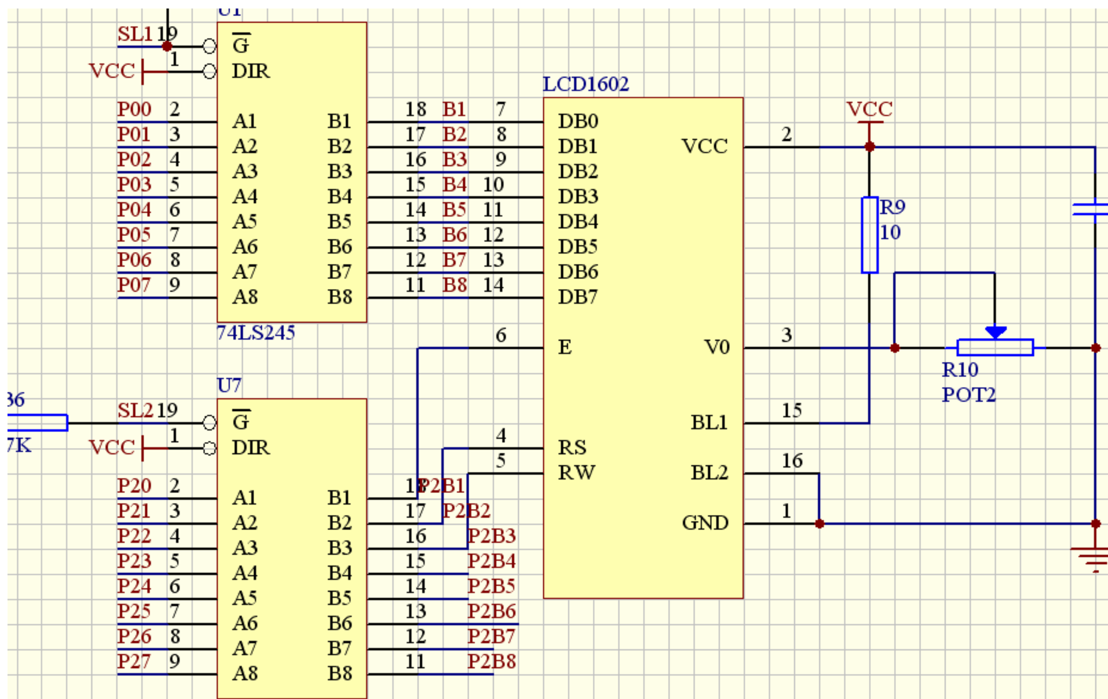
管脚	符号	名称	功能
1	VSS	接地	0V
2	VDD	电源	5V±10％
3	VEE	液晶驱动电压	
4	RS	寄存器选择信号	H:数据寄存器 L:指令寄存器
5	R/W	读/写信号	H:读 L:写
6	E	片选信号	
7-14	DB0-DB7	数据线	

2．寄存器选择功能

RS	R/W	操作
0	0	指令寄存器写入
0	1	忙标志和地址计数器读出
1	0	数据寄存器写入
1	1	数据寄存器读出

3．指令功能

指令	指令码										说明	执行时间（540KHZ）
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
读出 RAM 的值	1	1	D7	D6	D5	D4	D3	D2	D1	D0	从内部 RAM 读取资料（DDRAM/CGRAM/IRAM/GDRAM）	72us
清除显示	0	0	0	0	0	0	0	0	0	1	将 DDRAM 填满“20H”，并且设定 DDRAM 的地址计数器（AC）到“00H”	4.6ms
地址归位	0	0	0	0	0	0	0	0	1	X	设定 DDRAM 的地址计数器(AC）到“00H”，并且将游标移到开头原点位置；这个指令并不改变 DDRAM 的内容	4.6ms
进入点设定	0	0	0	0	0	0	0	1	I/D	S	指定在资料的读取与写入时，设定游标移动方向及指定显示的移位	72us
显示状态开/关	0	0	0	0	0	0	1	D	C	B	D=1：整体显示 ON C=1：游标 ON B=1：游标位置 ON	72us
游标或显示移位控制	0	0	0	0	0	1	S/C	R/L	X	X	设定游标的移动与显示的移位控制位元；这个指令并不改变 DDRAM 的内容	72us
功能设定	0	0	0	0	1	DL	X	0 RE	X	X	DL=1　（必须设为 1） RE=1：　扩充指令集动作 RE=0：　基本指令集动作	72us
设定 CGRAM 地址	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	设定 CGRAM 地址到地址计数器（AC）	72us
设定 DDRAM 地址	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	设定 DDRAM 地址到地址计数器（AC）	72us
读取忙碌标志（BF）和地址	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	读取忙碌标志（BF）可以确认内部动作是否完成，同时可以读出地址计数器（AC）的值	0us
写资料到 RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	写入资料到内部的 RAM（DDRAM/CGRAM/IRAM/GDRAM）	72us



注意：做本实验，需把拨码开关 K12 的 7，8 拨上去。

实例程序：

描述:1602 字符型 LCD 显示演示程序

在第一行显示　How are you

在第二行显示　www.ljd-2008.com

```
;端口定义
RS  EQU   P2.1
RW  EQU   P2.2
EP  EQU   P2.0
    ORG   0030H
    LJMP  MAIN
MAIN:
    LCALL LCD_INIT     ;初始化 LCD
    MOV   A,#15
```

```
    LCALL DELAY_MS      ;
MAIN_LOOP:                   ;  在第一行显示字符串"how are you"
    MOV    A,#03H
    LCALL SET_LCD_POS       ; 设置 LCD 光标到第一行的第 5 个字符
    MOV A ,#0FFH
    LCALL DELAY_MS
    MOV    DPTR,#TAB_WORD       ;"how are you"字串表格地址
    LCALL DISPLAY_STRING        ; 显示字符串
    MOV A ,#0FFH
    LCALL DELAY_MS ; 在第二行显示字符串"www.LJD-2008.COM"
    MOV    A,#40H            ;
    LCALL SET_LCD_POS       ;设置第二行第一个字符位置
    MOV A ,#0FFH
    LCALL DELAY_MS
    MOV    DPTR,#TAB_LJD
    LCALL DISPLAY_STRING
                                                ; 闪烁显示内容
    MOV    A,#200            ;
    LCALL DELAY_MS      ;
    LCALL LCD_TURN_OFF
    MOV    A,#200            ;
    LCALL DELAY_MS      ;
```

```
    LCALL LCD_TURN_ON
    MOV   A,#200          ;
    LCALL DELAY_MS     ;
    LCALL LCD_TURN_OFF
    MOV   A,#200          ;
    LCALL DELAY_MS     ;
    LCALL LCD_TURN_ON
    MOV   A,#200          ;
    LCALL DELAY_MS     ;
                                          ;清屏
    LCALL LCD_CLEAR
    MOV   A,#1
    LCALL DELAY_MS                    ; 重新显示
    JMP   MAIN

    ;显示字符串函数
    ;传入参数：DPTR(字符串表格地址)
    ;返回值：无
DISPLAY_STRING:
    CLR   A
    MOVC  A,@A+DPTR         ;
```

```
    JZ  END_DISPLAY_STRING      ; 如果遇到 00H 表示表格结束
    LCALL LCD_WRITE_DATA        ; 写数据到 LCD
    INC     DPTR              ; 指向表格的下一字符
    MOV     A, #100             ;
    LCALL DELAY_MS        ;
    SJMP   DISPLAY_STRING       ; 循环直到字符串结束
END_DISPLAY_STRING:
    RET
    ; 初始化 LCD
LCD_INIT:
    ; 设置显示格式---
    MOV     A,#38H              ; 38H --- 16*2 行显示,5*7 点阵,8 位数
据接口
    LCALL LCD_WRITE_COMMAND
    MOV     A,#1
    LCALL DELAY_MS
  ;开显示
    LCALL LCD_TURN_ON
  ;读写后指针加 1
    MOV     A,#06H              ; 06H ---  读写后指针加 1
    LCALL LCD_WRITE_COMMAND
    MOV     A,#1
```

```
    LCALL  DELAY_MS
  ; 清除 LCD 屏幕
    LCALL  LCD_CLEAR
    RET
 ;开显示
LCD_TURN_ON:
    MOV     A,#0CH              ; 0CH --- 开显示,无光标
    LCALL  LCD_WRITE_COMMAND
    MOV     A,#1
    LCALL  DELAY_MS
    RET
  ; 关显示
LCD_TURN_OFF:
    MOV     A,#08H              ; 08H --- 关显示
    LCALL  LCD_WRITE_COMMAND
    MOV     A,#1
    LCALL  DELAY_MS
    RET
; 清除 LCD 屏幕
LCD_CLEAR:
    MOV     A,#01H              ; 01H 清屏指令
    LCALL  LCD_WRITE_COMMAND
```

```
    MOV     A,#1
    LCALL  DELAY_MS
    RET
;设置 LCD 当前光标的位置
SET_LCD_POS:
    ORL     A,#80H          ;
    LCALL  LCD_WRITE_COMMAND
    RET
; 写入控制指令到 LCD
; 传入参数: ACC(要写入的命令)
; 返回值: 无
LCD_WRITE_COMMAND:
    ;LCALL    CHECK_LCD_BUSY
    CLR     RS
    CLR     RW
    CLR     EP
    NOP
    NOP
    MOV     P0,A            ; 写入数据到 LCD 端口
    NOP
    NOP
    NOP
```

```
    NOP
    SETB    EP
    NOP
    NOP
    NOP
    NOP
    CLR     EP
    RET
   ; 写入显示数据到 LCD
   ; 传入参数：ACC(要写入的数据)
   ; 返回值：无
LCD_WRITE_DATA:
    SETB    RS
    CLR     RW
    CLR     EP
    NOP
    NOP
    MOV     P0,A            ; 写入数据到 LCD 端口
    NOP
    ;NOP
    ;NOP
    ;NOP
```

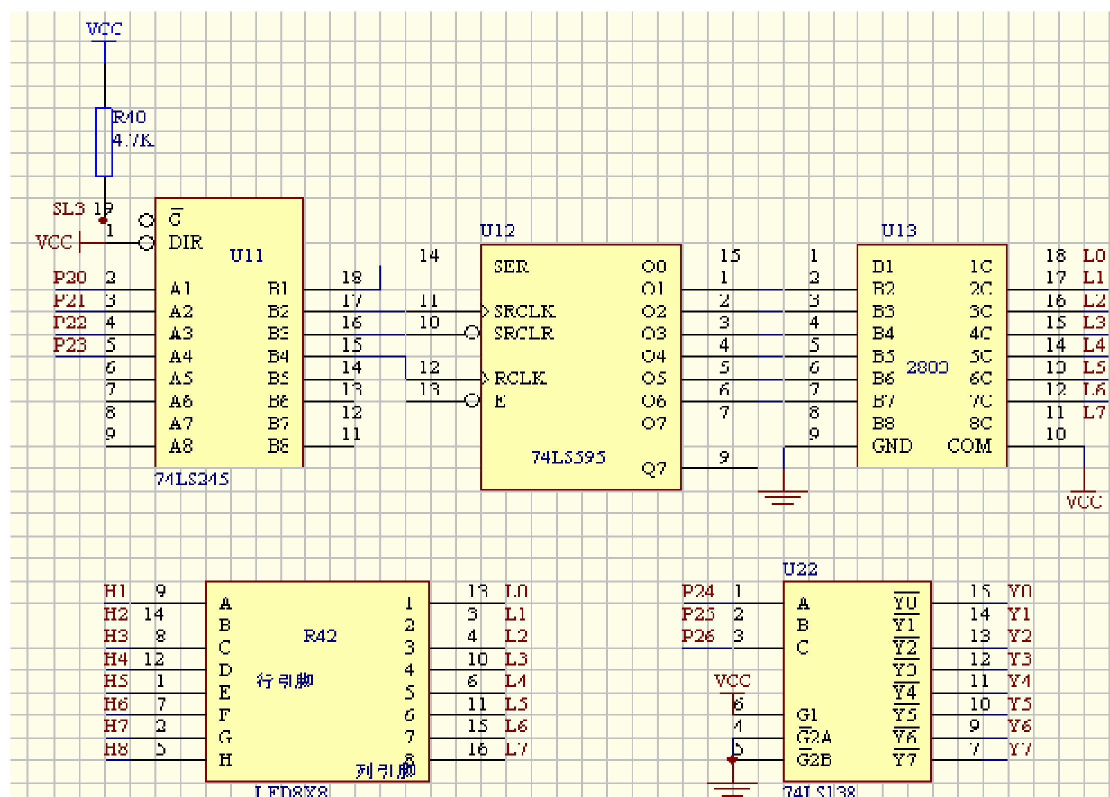
```
    SETB   EP
    NOP
    NOP
    NOP
    NOP
    CLR    EP
    RET
; 延时子程序
; 传入参数：ACC(延时时间,单位毫秒)
; 返回值：无
DELAY_MS:
    MOV    R7,A
DELAY_LOOP1:
    MOV    R6,#0FFH
DELAY_LOOP2:
    NOP
    NOP
    DJNZ   R6,DELAY_LOOP2
    DJNZ   R7,DELAY_LOOP1
    RET
TAB_LJD:
    DB "www.ljd-2008.com"
```

```
    DB 00          ; 字符结束标志
TAB_WORD:
    DB "How are you"
    DB 00          ; 字符结束标志
END
```

3－4 8×8 点阵输出

硬件设计与基本概念

LED 点阵在广告牌、字幕机及路况标识等中经常应用。常用的点阵规格有 5×7、5×8 及 8×8 等。



;***

描述：画面为沙漏

```
   ORG 0000H
ST: MOV DPTR,#TAB   ;显示数据表首址
LP5:CLR P2.2   ;74HC595 输出清零
    SETB P2.2
    CLR P2.4            ;选取第一行 000
    CLR P2.5
    CLR P2.6
    MOV  A,#00H         ;清累加器
    MOVC A,@A+DPTR      ;取显示数据
    LCALL DIS            ;送列数据子程?
    LCALL DELAY        ; 延时

    SETB P2.4            ;选取第二行 001
    CLR P2.5
    CLR P2.6
    INC DPTR
    MOV A,#00H
    MOVC A,@A+DPTR
    LCALL DIS
    LCALL DELAY
    CLR P2.4             ;选取第三行 010
    SETB P2.5
```

```
CLR P2.6
INC DPTR
MOV A,#00H
MOVC A,@A+DPTR
LCALL DIS
LCALL DELAY
 SETB P2.4              ;选取第四行 011
SETB P2.5
CLR  P2.6
INC DPTR
MOV A,#00H
MOVC A,@A+DPTR
LCALL DIS
LCALL DELAY
CLR P2.4                    ;选取第五行 100
CLR P2.5
SETB P2.6
INC DPTR
MOV A,#00H
MOVC A,@A+DPTR
LCALL DIS
LCALL DELAY
```

```
SETB P2.4            ;选取第六行 101
CLR P2.5
SETB P2.6
INC DPTR
MOV A,#00H
MOVC A,@A+DPTR
LCALL DIS
LCALL DELAY
CLR P2.4             ;选取第七行 110
SETB P2.5
SETB P2.6
INC DPTR
MOV A,#00H
MOVC A,@A+DPTR
LCALL DIS
LCALL DELAY
SETB P2.4           ;选取第八行 111
SETB P2.5
SETB P2.6
INC  DPTR
MOV  A,#00H
MOVC A,@A+DPTR
```

```
    LCALL DIS
    LCALL DELAY
    JMP ST              ;一桢画面显示完成返回继续显示下一祯
;**********************************************************
;送列数据子程序
;**********************************************************
DIS:CLR C
LP1:MOV R5,#08H
LP: RRC A
    MOV P2.0,C
    CLR P2.1
    SETB P2.1
    DJNZ R5, LP
    CLR P2.3
    SETB P2.3
    NOP
    NOP
    RET
;**********************************************************
;延时子程序
;**********************************************************
DELAY: MOV R5,#04H
```

```
H0:    MOV R6,#0ffH
H1 :   MOV R7,#0ffH
H2:    DJNZ R7,H2
       DJNZ R6,H1
       DJNZ R5,H0
       RET
;***********************************************************
;显示数据表
;***********************************************************
TAB:
DB  0ffH,7eH,3cH,18H,18H,3cH,7eH,0FFH
TAB1:
DB  10,0feH,92H,92H,0feH,92H,10H,10H
    END
```

3－5　I2C（24C02）读、写实验

1、串行EEPROM（24C02）接口方法

在新一代单片机中，无论总线型还是非总线型单片机，为了简化系统结构，提高系统的可靠性，都推出了芯片间的串行数据传输技术，设置了芯片间的串行传输接口或串行总线。串行总线扩展接线灵活，极易形成用户的模块化结构，同时将大大简化其系统结构。串行器件不仅占用很少的资源和 I/O 线，而且体积大大缩小，同时还具有

工作电压宽，抗干扰能力强，功耗低，数据不宜丢失和支持在线编程等特点。目前，各式各样的串行接口器件层出不穷，如：串行 EEPROM，串行 ADC/DAC，串行时钟芯片，串行数字电位器，串行微处理器监控芯片，串行温度传感器等等。

串行 EEPROM 是在各种串行器件应用中使用较频繁的器件，和并行 EEPROM 相比，串行 EEPROM 的数据传送的速度较低，但是其体积较小，容量小，所含的引脚也较少。所以，它特别适合于需要存放非挥发数据,要求速度不高,引脚少的单片机的应用。这里介绍串行 EEPROM 芯片，以及它们和单片机的接口技术。

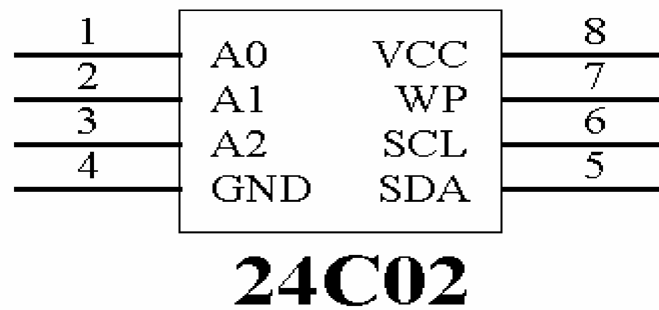
2、串行 **EEPROM** 及其工作原理

串行 EEPROM 中，较为典型的有 ATMEL 公司的 AT24CXX 系列以及该公司生产的 AT93CXX 系列，较为著名的半导体厂家，包括 Microchip，国家半导体厂家等，都有 AT93CXX 系列 EEPROM 产品。

AT24CXX 系列 **EEPROM**

AT24CXX 系列的串行电可改写及可编程只读存储器 EEPROM 有 10 种型号，其中典型的型号有 AT24C01A/02/04/08/16 等 5 种，它们的存储容量分别是 1024/2048/4096/8192/16384 位，也就是 128/256/512/1 024/2048 字节。这个系列一般用于低电压，低功耗的工业和商业用途，并且可以组成优化的系统。这个系统还有多种包括 5V(4.5~5.5V),2.7V(2.7~5.5V),2.5V(2.5~5.5V),1.8V(1.8~5.5V) 等 4 种电压级别。它们的封装有 8 引脚 PDIP 方式，8 引脚和 16 引脚 SOIC 方式。信息存取采用 2 线串行接口。这里我们就 24C02 的结构特点，其他系列比较类似。

3、结构原理及引脚



AT24C02 有地址线 A0~A2，串行数据引脚 SDA，串行时钟输入引脚 SCL，写保护引脚 WP 等引脚。很明显，其引脚较少，对组成的应用系统可以减少布线，提高可靠性。

各引脚的功能和意义如下。

①VCC 引脚，电源+5V。

②GND 引脚，地线。

③SCL 引脚，串行时钟输入端。在时钟的正跳沿即上升沿时把数据写入 EEPROM；在时钟的负跳沿即下降沿时把数据从 EEPROM 中读出来。

④SDA 引脚，串行数据 I/O 端，用于输入和输出串行数据。这个引脚是漏极开路的端口，故可以组成“线或”结构。

⑤A0,A1,A2 引脚，是芯片地址引脚。在型号不同时意义有些不同，但都要接固定电平。

⑥WP 引脚，写保护端。这个端提供了硬件数据保护。当把 WP 接地时，允许芯片执行一般读写操作；当把 WP 接 VCC 时，则对芯片实施写保护。

⑦NC 引脚，无用引脚，不接任何电平。

4、存储器的组织及运行

①存储器的组织：对于不同的型号，存储器的组织不一样，其关键原因在于存储器容量存在差异。对于 AT24CXX 系列的 EEPROM，其典型型号的存储器组织如下。

AT24C01A:内部含有 128 个字节,故需要 7 位地址对其内部字节进行寻址

AT24C02:内部含有 256 个字节，故需要 8 位地址对其内部字节进行读写。

5、运行方式：

对于时钟及数据传送,串行数据 I/O 端口 SDA 一般需要用外部上拉电阻将其电平拉高。加到 SDA 的数据只有在串行时钟 SCL 对于低电平的时间周期内可以改变。当串行时钟 SCL 处于高电平时，SDA 的数据变化用于指明起始或停止状态。在 SCL 为高电平期间，如果 SDA 从低电平上升到高电平，则表示起始状态；如果 SDA 从高电平下降到低电平，则表示停止状态。

起始状态：当 SCL 为高电平时，SDA 由高电平变到低电平则处于起始状态。起始状态应处于任何其他命令之前。

停止状态：当 SCL 处于高电平时，SDA 从低电平变到高电平则处于停止状态。在执行完读序列信号之后，停止命令将把 EEPROM 置于低功耗的备用方式(Standby Mode).

应答信号：应答信号是由接受数据的器件发出的。当 EEPROM 接受完一个写入数据之后，会在 SDA 上发一个”0”应答信号。反之，当单片机接受完来自 EEPROM 的数据后，单片机也应向 SDA 发 ACK 信号。

ACK 信号在第 9 个时钟周期时出现。

备用方式(Standby Mode)：AT24C01A/02/04/08/16 都具有备用方式，以保证在没有读写操作时芯片处于低功耗状态。在下面两种情况中，EEPROM 都会进入备用方式：第一，芯片通电的时候；第二，在接到停止位和完成了任何内部操作之后。

AT24C02 等 5 种典型的 EEPROM 在进入起始状态之后，需要一个 8 位的“器件地址字”去启动存储器进行读或写操作。在写操作中，它们有“字节写”,“页面写”两种不同的写入方法。在读操作中，有“现行地址读”，随机读和“顺序读”种各具特点的读出方法。下面分别介绍器件寻址，写操作和读操作。

① 器件寻址：所谓器件寻址(Device Addressing)就是用一个 8 位的器件地址字(Device Address Word)去选择存储器芯片。在逻辑电路中的 AT24CXX 系列的 5 种芯片种，即 AT24C01A/02/04/08/16 中，如果和器件地址字相比较结果一致，则读芯片被选中。下面对器件寻址的过程和意义加以说明。

② 芯片的操作地址

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	A2	A1	A0	R/W

用于存储器 EEPROM 芯片寻址的器件地址字如图所示。它有 4 种方式，分别对应于 1K/2K,4K,8K 和 16K 位的 EEPROM 芯片。

从图中看出：器件地址字含有 3 个部分。第一部分是高 4 位，它们称为 EEPROM AT24C01A/02/04/08/16 的标识第二部分称为硬布线地址，它们是标识后的 3 位。第三部分是最低位，它是读/写操作选择位。

第一部分：器件标识，器件地址字的最高 4 位。这 4 位的内容恒为 ” 1010 ”，用于标识 EEPROM 器件 AT24C01A/02/04/08/16。

第二部分：硬布线地址，是与器件地址字的最高 4 位相接的低 3 位。硬布线地址的 3 位有 2 种符号：Ai(i=0~2),Pj(j=0~2)其中 Ai 表示外部硬布线地址位

对于 AT24C10A/02 这两种 1K/2K 位的 EEPROM 芯片,硬布线地址为 ” A2,A1,A0 ” .在应用时，” A2,A1,A0 ” 的内容必须和 EEPROM 芯片的 A2,A1,A0 的硬布线情况，即逻辑连接情况相比较，如果一样，则芯片被选中；否则，不选中。

AT24C01A/02:真正地址=字地址

第三部分：读/写选择位，器件地址字的最低位，并用 R/W 表示。当 R/W=1 时，执行读操作；当 R/W=0 时，执行写操作。

当 EEPROM 芯片被选中时，则输出 ” 0 ”; 如果 EEPROM 芯片没有被选中，则它回到备用方式。被选中的芯片。其以后的输入，输出情况视写入和读出的内容而定。

②写操作：AT24C01A/02/04/08/16 这 5 种 EEPROM 芯片的写操作有 2 种：一种是字节写，另一种是页面写。

字节写

这种写方式只执行 1 个字节的写入。字节写的过程如图所示，其写入过程分外部写和内部写两部分，分别说明如下。

在起始状态中，首先写入 8 位的器件地址。则 EEPROM 芯片会产生一个”0”信号 ACK 输出作为应答；接着，写入 8 位的字地址，在接受了字地址之后，EEPROM 芯片又产生一个”0”应答信号 ACK；随后，写入 8 位数据，在接受了数据之后，芯片又产生一个”0”信号 ACK 作为应答。到此为止，完成了一个字节写过程，故应在 SDA 端产生一个停止状态，这是外部写过程。

在这个过程中，控制 EEPROM 的单片机应在 EEPROM 的 SCL，SDA 端送入恰当的信号。当然在一个字节写过程结束时，单片机应以停止状态结束写过程。在这时，EEPROM 进入内部定时的写周期，以便把接受的数据写入到存储单元中。在 EEPROM 的内部写周期中，其所有输入被屏蔽，同时不响应外部信号直到写周期完成。这是内部写过程。内部写过程大约需要 10ms 时间。内部写过程处于停止状态与下一次起始状态之间，页面写

这种写入方式执行含若干字节的 1 个页面的写入。对于 AT24C01A/02，它们的 1 个页面含 8 个字节；页面写的开头部分和字节写一样。在起始状态，首先写入 8 位器件地址；待 EEPROM 答当了”0”信号 ACK 之后，写入 8 位字地址；又待芯片应答了”0”信号 ACK 之后，写入 8 位数据。

随后页面写的过程则和字节写有区别。

当芯片接受了第一个 8 位数据并产生应答信号 ACK 之后，单片机

可以连续向 EEPROM 芯片发送共为 1 页面的数据。对于 AT24C01A/02，可发送共 1 个页面的 8 个字节（连第一个 8 位数据在内）。对于 AT24C04/08/16，则共可发送 1 个页面共 16 个字节（连第一个 8 位数据在内）。当然，每发一个字节都要等待芯片的应答信号 ACK。

之所以可以连续向芯片发送 1 个页面数据，是因为字地址的低 3~4 位在 EEPROM 芯片内部可实现加 1，字地址的高位不变，用于保持页面的行地址。页面写和字节写两者一样可，都分为外部写和内部写过程。

应答查询：应答查询是单片机对 EEPROM 各种状态的一种检测。单片机查询到 EEPROM 有应答 ”0 ”信号 ACK 输出，则说明其内部定时写的周期结束，可以写入新的内容。单片机是通过发送起始状态及器件地址进行应答查询的。由于器件地址可以选择芯片，则检测芯片送出到 SDA 的状态就可以知道其是否有应答了。

③读操作：读操作的启动是和写操作类同的。它一样需要图所示的器件地址字。和写操纵不同的就是信号为时执行读操作。

读操纵有 3 种方式，即现行地址读，随机读和顺序读。下面分别说明它们的工作过程。

现行地址读

在上次读或写操纵完成之后。芯片内部字地址计数器会加 1，产生现行地址。只要没有再执行读或写操作，这个现行地址就会在 EEPROM 芯片保持接电的期间一直保存。一旦器件地址选中 EEPROM 芯片，并且有 R/W=1，则在芯片的应答信号 ACK 之后把读出的现行地址

的数据送出。现行地址的数据输出时，就由单片机一位一位接受，接收后单片机不用向 EEPROM 发应答信号 ACK ”0”电平，但应保证发出停止状态的信号以结束现行地址读操作。现行地址读会产生地址循环覆盖现象，但和写操纵的循环覆盖不同。在写操纵中，地址的循环覆盖是现行页面的最后一个字节写入之后，再行写入则覆盖同一页面的第一个字节。而在现行地址读操纵中，地址的循环覆盖是在最后页面的最后一个字节读出之后，再行读出才覆盖第一个页面的第一个字节。

随机读

随机读和现行地址读的最大区别在于随机读会执行一个伪写入过程以把字地址装入 EEPROM 芯片中，然后执行读出，

显然，随机读有 2 个步骤。

1. 执行伪写入——把字地址送入 EEPROM，以选择需读的字节。

2. 执行读出——根据字地址读出对应内容。

当 EEPROM 芯片接收了器件地址及字地址时，在芯片产生应答信号 ACK 之后，单片机必须再产生一个起始状态，执行现行地址读，这时单片机再发出器件地址并且令 R/W=1，则 EEPROM 应答器件地址并行输出被读数据。在数据读出时由单片机执行一位一位接收，接收完毕后，单片机不用发”0”应答信号 ACK，但必须产生停止状态以结束随机读过程。

应该注意：在随机读的第二个步骤是执行现行地址读的，由于第一个步骤时芯片接收了字地址，故现行地址就是所送入的字地址。

顺序读

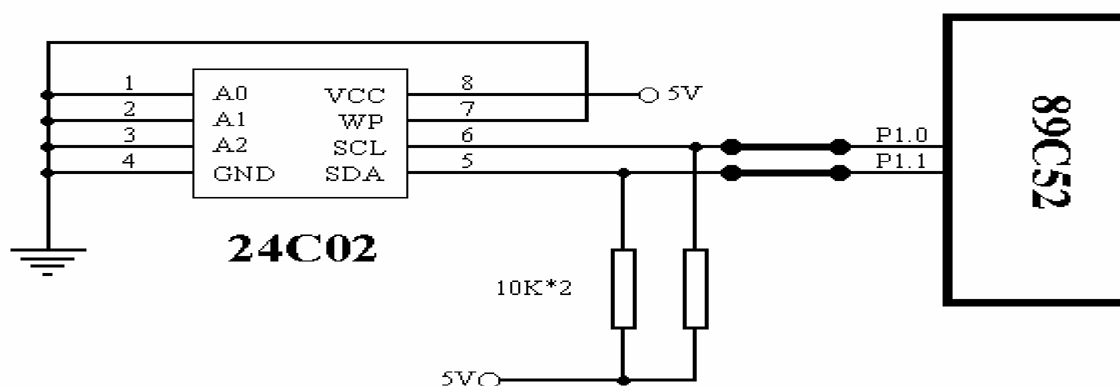
顺序读可以用现行地址读或随机读进行启动。它和现行地址读。随机读的最大区别在于：顺序读在读出一批数据之后才由单片机产生停止状态结束读操作；而现行地址读和随机读在读出一个数据之后就由单片机产生停止状态结束读操作。

执行顺序读时，首先执行现行读或随机读的有关过程，在读出第一个数据之后，单片机输出“0”应答信号 ACK。在芯片接收应答信号 ACK 后，就会对字地址进行计数加 1，随后串行输出对应的字节。当字地址计数达到存储器地址的极限时，则字地址会产生覆盖，顺序读将继续进行。只有在单片机不再产生“0”应答信号 ACK，而在接收数据之后马上产生停止状态，才会结束顺序读操作。

在对 AT24CXX 系列执行读写的 2 线串行总线工作中，其有关信号是由单片机的程序和 EEPROM 产生的。有两点特别要记住：串行时钟必须由单片机程序产生，而应答信号 ACK 则是由接收数据的器件产生，也就是写地址或数据时由 EEPROM 产生 ACK，而读数据时由单片机产生。

（4）AT24CXX 系列应用注意事项

AT24CXX 系列型号：AT24CXX 系列 EEPROM 有 13 种型号。它们的容量不同，执行页历写时的页历定义不同，进行读写时的地址位数也不同，器件地址不同。有关主要指标在应用中要加以区别和注意。



实验程序

1、写把 CPU 的 RAM　31H～38H 的数据送到 24C02 的 00H～07H 单元

CPU 写数据到 24C02 的数据格式如下:

S	从控制器的地址+W	A	写入单元地址	A	DATA 1	A	DATA2	A	…	DATA N	A	P

其中：S 表起始条件，　A 表示应答响应，P 表示停止条件

```
        ;   文件名称:W24C02.ASM
            SCL EQU P1.0
            SDA EQU P1.1
               ORG 0000H
WR_EEROM:      MOV R6,#40H
            MOV 30H,#00H
            MOV R0,#30H
W_LOOP:        LCALL WR_DATA
W_LOOP2:  INC @R0
```

```
        DJNZ R6,W_LOOP1
        SJMP STOP
W_LOOP1: LCALL WR_DATA2
        SJMP W_LOOP2
STOP:    LCALL STOP24
        SJMP $
;--------------------------------------
WR_DATA:    LCALL START
        MOV A,#0A0H
        LCALL WBYTE
WR_DATA1:   MOV A,@R0
        LCALL WBYTE
WR_DATA2:   MOV A,#99H
            LCALL WBYTE
        RET
;--------------------------------------
WBYTE:      NOP
        MOV R3,#08H
WBY0:    CLR SCL
        RLC A
        MOV SDA,C
        SETB SCL
```

```
        DJNZ R3,WBY0
        CLR SCL
        NOP
        SETB SCL
        NOP
        JB SDA,$
        CLR SCL
        NOP
        RET
;---------------------------------
START:  CLR     SCL
        NOP
        SETB  SDA
        NOP
        SETB  SCL
        NOP
        CLR     SDA
        NOP
        CLR     SCL
        RET
;---------------------------------
STOP24: CLR     SCL
```

```
        NOP
        CLR      SDA
        NOP
        SETB  SCL
        NOP
        SETB  SDA
        NOP
        CLR   SCL
        RET
END
```

２、读 24C02 的 00H～07H 到 CPU 的 RAM　31H～38H 单元。

CPU 从 24C02 内读数据格式如下:

S	从控制器的地址+W	A	写入读单元地址	S	从控制器的地址+R	A	Data1	A	··	Data n	A	P

其中: S 表起始条件， A 表示应答响应，P 表示停止条件

```
;文件名称:R24C02.ASM
            SCL EQU P1.0
            SDA EQU P1.1
            ORG 0000H
RD_EEROM:   MOV R6,#08H
            MOV 30H,#00H
```

```
          MOV R0,#30H
          MOV R1,#40H
R_LOOP:   LCALL START
          MOV A,#0A0H
          LCALL WBYTE
          MOV A,@R0
          LCALL WBYTE
R_LOOP1:  LCALL START
          MOV A,#0A1H
          LCALL WBYTE
          LCALL RBYTE
          MOV @R1,A
          INC R1
          DJNZ R6,R_LOOP1
          LCALL STOP24
          SJMP  $
;-------------------------------------------
RBYTE:    NOP
          MOV R3,#08H
RBY0:     CLR SCL
          NOP
          SETB  SCL
```

```
        NOP
        MOV C,SDA
        RLC A
        DJNZ R3,RBY0
        CLR SCL
        NOP
        SETB SDA
        NOP
        SETB SCL
        RET
;---------------------------------------------
WBYTE:  MOV R3,#08H
WBY0:   CLR SCL
        NOP
        RLC A
        MOV SDA,C
        SETB SCL
        DJNZ R3,WBY0
        CLR SCL
        NOP
        SETB SCL
        NOP
```

```
        JB SDA,$
        CLR SCL
        NOP
        RET
;---------------------------------------------
START:  CLR   SCL
        NOP
        SETB  SDA
        NOP
        SETB  SCL
        NOP
        CLR   SDA
        NOP
        NOP
        NOP
        CLR   SCL
        RET
;---------------------------------
STOP24:    CLR   SCL
        NOP
        CLR   SDA
        NOP
```

```
        SETB  SCL
        NOP
        SETB  SDA
        NOP
        CLR   SCL
        RET
END
```

3－6 八位串行输出 **A/D** 转换器 **TLC549** 及应用程序

1.1 硬件描述

TLC549 是以八位开关电容逐次逼近 A/D 转换器为基础而构造的 CMOS A/D 转换器。其设计能通过三态数据输出和模拟输入与微处理器或外围设备串行接口。TLC549 仅用输入/输出时钟（CLK）和芯片选择(CS)输入做数据控制。TLC548 的最高 CLK 输入频率为 2.048MHZ，而 TLC549 的最高 CLK 输入频率为 1.1 MHZ.

TLC548/549 的内部提供了片内系统时钟,它通常工作在 4 MHZ 且不需要外部元件.片内系统时钟使内部器件的操作独立于串行输入/输出的时序,并允许 TLC548/549 像许多软件和硬件要求的那样工作.CLK 和内部系统时钟一起,可以实现高速数据传送以及对 TLC548 为每秒 45 500 次转换、对 TLC549 为每秒 40 000 次的转换。

TLC548/549 为了提高灵活性和访问速度，设有两个控制输入端（时钟 CLK 和片选 CS)。这些控制输入和与 TTL 兼容的三态输出易于

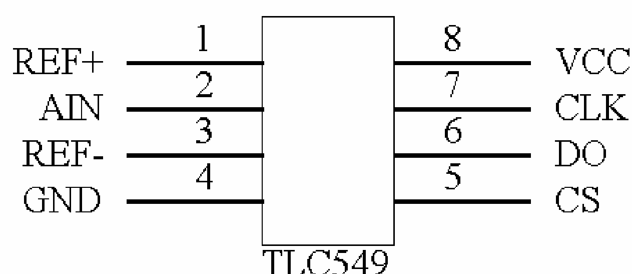
与微处理器或小型计算机串行通信。器件可在较短的时间内完成转换。TLC548 每 22μ□重复一次完整的输入—转换—输出周期，TLC549 每 25μ□重复一次输入—转换—输出周期。

内部时钟和 CLK 独立使用，且不需要任何特定的速度或二者之间的相位关系。这种独立性简化了硬件和软件控制任务。由于这种独立性和系统时钟的内部产生，控制硬件和软件只需关心利用 I/O 时钟读出先前转换结果和启动转换。

TLC548/549 的其它特点包括通用控制逻辑，可自动工作或在微处理器控制下工作，且有片内采样保持电路，具有差分高阻抗基准电压输入端，易于实现定标以及与逻辑电路和电源噪声的隔离；整个开关电容逐次逼近转换器的设计，允许在小于 1 7μ□的时间内以最大总误差为±0.5 最低有效位（LSB）的精度实现转换；电源范围为+3~+6V，功耗小于 15mW；能理想地用于包括电池供电便携式仪表的低成本、高性能系统中。

TLC548/549C 的工作温度范围为 0°C~70°C，TLC548I/549I 的工作温度范围为-40°C~85°C。TCL548C/549C 的引脚和控制信号与 TLC540 八位 A/D 转换器以及 TLC1540 10 位 A/D 转换器兼容。

1、 引脚排列



2、

TLC548/549 的引脚排列如图 4.2 所示。其中，基准（REF+，REF-）为差分输入，可以将 REF-接地，REF+接 VCC 端，但要加滤波电容。大于加至 REF 电压的模拟输入电压转换为全“ 1 ”，小于加至 REF 电压的模拟输入电压转换为全 “ 0 ”。为了工作良好，REF+电压应比 REF-电压至少高 1V。而且，当此差分基准电压降至 4.75V 以下时，总失调误差可能增加。

3、 操作说明

TLC548/549 的 CS 为高电平时，DATA OUT 处于高阻态且 CLK（I/O 时钟）被禁止。当使用另外的 TLC548 或 TLC549 器件时，这种 CS 控制功能允许 CLK 与其共用同一时钟。当使用多个 TLC548 和 TLC549 器件时，这也使所需的控制逻辑为最少。该器件的工作时序如图 4.3 所示。其正常的控制时序如下。

（1）CS 被拉至低电平。为了使 CS 端噪声所产生的误差为最小，在识别低跳变之前，内部电路在 CS 之后等待内部系统时钟两个上升沿与其后的下降沿.然而,由于 CS 上升沿的作用,即使直到经历了一段时间,其余的集成电路仍不识别跳变.DATA OUT 也将在较短时间之内变为高阻状态.当器件用于噪声环境中时,这种技术可用来保护器件使其免受噪声的影响.当 CS 变为低电平时,前次转换结果的最高有效位(MSB)开始出现在 DATA OUT 端。

（2）前四个 CLK 周期输出前次转换结果的第七、六、五、四、三位。在 CLK 第四个高电平至低电平的跳变之后，片内采样和保持电路开始对模拟输入采样。采样操作主要是将模拟输入信号充到内部电

容器上。

（3）其后再把三个时钟送至 CLK 端，在这些时钟周期的下降沿，第二、一、0 位被移出。

（4）最后一个（第八个）时钟周期被加至 CLK 端。此时钟周期高电平至低电平的跳变使片内采样和保护电路开始保持。保持功能在接着四个内部系统时钟后结束，且在下面 32 个内部时钟周期内完成转换，总共为 36 个周期。在第八个 CLK 周期之后，CS 必须变为高电平，否则，CLK 必须保持低电平至少 36 个内部系统时钟周期，以供保持和转换功能的完成。在多个转换周期内使 CS 保持低电平，必须特别注意防止 CLK 线上的噪声闪变。如在 CLK 上发生闪变，那么在微处理器和器件之间的 I/O 时序将失去同步。此外，如果 CS 为高电平，那么它必须保持高电平直至转换结束为止；否则，CS 的有效高电平至低电平跳变将引起复位，使正在进行的转换失败。

在 36 个系统时钟周期发生之前，通过完成以上步骤（1）至（4）可以启动新的转换，同时正在进行的转换中止。所读取的是先前的转换结果而不是正在进行的转换结果。

对于某些应用，诸如选通应用，需要在特定的时间点启动转换。TLC548/549 可以实现之。虽然片内采样和保持在第四个有效 CLK 时钟周期的负沿开始采样，但是直到第八个有效 I/O 时钟周期的负边之前，保持功能并不开始。它应当开始于必须转换模拟信号的瞬间。TCL548/549 继续采样模拟输入，到 I/O 时钟的第八个下沿为止。然后，控制电路或软件立即拉低 I/O CLK 并启动保持功能以及在所需时

间点保持模拟信号并开始转换。

4、 应用电路

TLC548/549 与 CPU 的接口很简单。只要将 TLC548/549 的 DO、CLK 和 CPU 的串口 RXD、TXD 相连，CS 和 CPU 的 I/O 口相接即可，以适应 TLC548/549 的单极性要求。A/D 直接与 89C51 连接，编程采用串口方式。

通常在正常的室温附近，对于远程数据采集（传感器和 A/D 转换器放在离微处理器几英尺远处），在输入时钟跳变时间慢至 2μS 的情况下，TLC548/549 仍可正常工作。为了使 CS 端噪声所引起的误差最小，在响应控制输入信号以前，器件内部电路在 CS 之后等待内部系统时钟两个上升沿和一个下降沿。这样可以确保器件工作的可靠性。

```
;**********************************************************
;标题:北京启东微芯 LJD-SY-5100 单片机实验 549A/D 转换严示程序
;文件:TLC549.asm
;描述:
通过数码管观看 A/D 采集的数据如显示异常请按复位键
;**********************************************************
CS          BIT      P1.5
LOAD        BIT      P1.2
DIN         BIT      P1.0
DCLK        BIT      P1.1
```

```
DOUT        BIT       P3.3
;***********************************
AD_DATA     EQU       036H       ;采集数据缓冲区
TIMER       DATA      030H
TIMER1      DATA      031H
TIMER2      DATA      032H
TIMER3      DATA      033H
;************************************
MAIN:        MOV P1,#61H            ;禁用其它芯片
TCL549:      MOV R3,#08H            ;计数器
             CLR CS                 ;开启芯片
             ACALL DELAY            ;延时
READ:        CLR DCLK               ;读 549 芯片数据
             NOP
             MOV C,DIN
             SETB DCLK
             RLC A
             DJNZ R3,READ
             MOV AD_DATA,A
             LCALL DELAY_1S        ;延时
             SJMP START            ;调数码管显示程序
;**********************************************
```

```
;延时子程序
;*********************************************
DELAY:MOV R5,#01H
 H0:  MOV R7,#0FFH
 H1:  DJNZ R7,H1
      DJNZ R5,H0
      RET
;**********************************
;数码管显示程序
;**********************************
START:
        SETB CS                  ;禁用 549 芯片
        NOP
        MOV P1,#60H             ;禁止其它芯片
        CLR     DIN              ;初始化 CH451
        SETB    DCLK
        SETB    DIN
        SETB    LOAD
        SETB    DOUT
        NOP
        MOV     B,#04H            ;设置 CH451
        MOV     A,#03H            ;关看门狗开显示键盘
```

```
        NOP
        LCALL   WRITE
        NOP
        LCALL DELAY
            MOV     B,#08H          ;显示位置
START1:     MOV     R5,#00H         ;加载字数据 1
TT1:        LCALL   ZHUAN
            LCALL   WRITE          ;将数据送入数码管显示
            LCALL   DELAY_1S
            INC     R5
            INC B                  ;显示位置加一
            CJNE    R5,#02H,TT1
            LCALL DELAY_1S
            JMP     MAIN           ;返回显示采集的数据
TAB:
        DB      03FH        ;0
        DB      006H        ;1
        DB      05BH        ;2
        DB      04FH        ;3
        DB      066H        ;4
        DB      06DH        ;5
        DB      07DH        ;6
```

```
        DB          07H                 ;7
        DB          07FH                ;8
        DB          06FH                ;9
        DB          77H                 ;A
        DB          07CH                ;B
        DB          039H                ;C
        DB          5EH                 ;D
        DB          079H                ;E
        DB          071H                ;F
 ;****************************************
 ;写数据到 ch451
 ;*****************************************
WRITE:
        PUSH        ACC
        CLR         EX0
        CLR         LOAD
        MOV         R7,#08H
WRITE_1:
        RRC         A
        CLR         DCLK
        MOV         DIN,C
        SETB        DCLK
```

```
        DJNZ        R7,WRITE_1
        MOV         A,B
        MOV         R7,#004H
WRITE_2:
        RRC         A
        CLR         DCLK
        MOV         DIN,C
        SETB        DCLK
        DJNZ        R7,WRITE_2
        SETB        LOAD
        SETB        EX1
        POP         ACC
        RET
;*********************************
DELAY_1S:
                MOV         TIMER1,#1
TEST_DYA:       MOV         TIMER2,#255
TEST_DYA1:      MOV         TIMER3,#255
TEST_DYA2:      NOP
         NOP
         DJNZ       TIMER3,TEST_DYA2
         DJNZ       TIMER2,TEST_DYA1
```

```
        DJNZ      TIMER1,TEST_DYA
        RET
;************************************
;将采集数据拆为为半字节
;*************************************
ZHUAN: MOV A,AD_DATA
       SWAP A
       MOV AD_DATA,A
       ANL A,#0FH    ;屏蔽高四位
       MOV      DPTR,#TAB
       MOVC     A,@A+DPTR
       RET
       END
```

3－7　10 位串行 D/A 转换器 TLC5615 及应用程序

1．1 硬件描述

TLC5615 是带有缓冲基准输入（高阻抗）的 10 位电压输出数字—模拟转换器（DAC）。DAC 具有基准电压两倍的输出电压范围，且 DAC 是单调变化的。器件可在单 5V 电源下工作，且具有上电复位功能以确保可重新启动。TLC5615 的数字控制通过三线串行总线进行，它与 CMOS 兼容且易于和工业标准的微处理器及单片机接口。器件接收 16 位数据字以产生模拟输出。数字输入端的特点包括带有斯密特触发

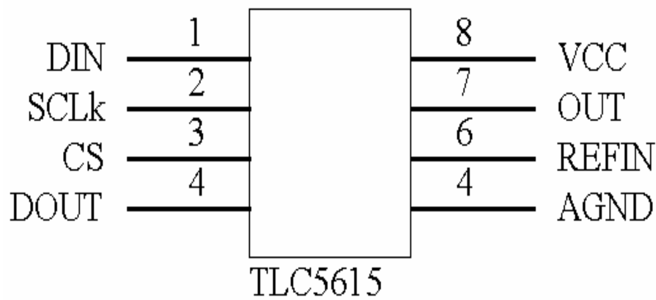
器，具有高噪声抑制能力。数字通信协议包括 SPI、QSP 以及 Microwire 标准。低功耗，在 5V 供电时功耗仅 1.75W；数据更新速率为 1.2MHZ；典型的建立时间为 12.5μS。

TLC5615 可广泛应用于电池供电测试仪表、数字增益调整、电池远程工业控制和移动电话等领域。

该器件外形为八脚小型 D 或 DIP 封装。C 档的工作温度范围为 0℃~70℃，I 档的工作温度范围为-40℃~85℃。其引脚与 Maxim 公司的 MAX515 完全兼容。

1、　　引脚排列

TLC5615 的引脚排列见图



在图 4.10 中，引脚 DOUT 用于菊花链的串行数据输出；DIN 是串行数据输入；SCLK 是串行时钟输入；CS 是选片端，低电平有效；OUT 是 D/A 电压输出；REFIN 是基准输入端，一般接 2V 到 Vcc—2V，典型值是 2.048V；Vcc 是电源端，一般接+5V；AGND 是模拟地。

2、　　使用说明

TLC5615 通过固定增益为 2 的运放缓冲电阻串网络，把 10 位数字数据转换为模拟电压电平。上电时，内部电路把 DAC 寄存器复位为

0。其输出具有与基准输入相同的极性，表达式为

VO=2 X REF X CODE/1024

1） 数据输入

由于 DAC 是 12 位寄存器，所以在 10 位数据字中必须写入数值为 0 的两个低于 LSB（D0）的位（次最低有效位）。

2） D/A 输出

输出缓冲器具有满电源电压幅度输出，它带有短路保护并能驱动有 100pF 负载电容的 2kΩ负载。

3） 外部基准

基准电压输入经过缓冲，这使得 DAC 输入电阻与代码无关。因此，REFIN 输入电阻为 10MΩ，REFIN 输入电容的典型值为 5pF，它们与输入代码无关。基准电压决定 DAC 的满度输出。

4） 逻辑接口

逻辑输入端可使用 TTL 或 CMOS 逻辑电平。但使用满电源电压幅度，CMOS 逻辑可得到最小的功耗。当使用 TTL 逻辑电平时，功率需求增加约两倍。

5） 串行时钟和更新速率

图 4。11 示出了 TLC5615 的工作时序。最大串行时钟速率近似为 14MHZ。通常，数字更新速率受片选周期限制。对于满度输入阶跃跳变，10 位 DAC 建立时间为 12.5μS，这把更新速率限制至 80kHz。

6） 菊花链接器件

假如时序关系合适，可以通过在一个链路（Chain）中把一个器件的

DOUT 端连接到下一个器件的 DIN 端实现 DAC 的菊花链接（级联）。DIN 处的数据延迟 16 个时钟周期加一个时钟宽度后出现在 DOUT 端。DOUT 是低功率的推拉输出电路。当 CS 为低电平时，DOUT 在 SCLK 下降沿变化；当 CS 为高电平时，DOUT 保持在最近数据位的值并不进入高阻状态。

3、　典型接口

当片选 CS 为低电平时，输入数据读入 16 位移位寄存器（由时钟同步，最高有效位在前）。SLCK 输入的上升沿把数据移入输入寄存器。接着，CS 的上升沿把数据传送至 DAC 寄存器。当 CS 为高电平时，输入数据不能由时钟同步送入输入寄存器。所有 CS 的跳变应当发生在 SCLK 输入为低电平时。

如果不使用菊花链（级联）功能，那么可以使用 MSB 在前的 12 位输入数据序列：

D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0

如果使用菊花链（级联）功能，那么可以传送四个高虚拟位在前的 16 位输入数据序列：

4 Upper Dummy	10 Data Bits	0	0

来自 DOUT 的数据需要输入时钟 16 个下降沿，因此，需要额外的时钟宽度。当菊花链接（级联）多个 TLC5615 器件时，因为数据传送需要 16 个输入时钟周期加上一个额外的输入时钟下降沿使数据在 DOUT 端输出，所以，数据需要四个高虚拟位（Upper Dummy Bits）。为了提供与 12 位数据转换器传送的硬件与软件兼容性，两个额外位

总是需要的。

TLC5615 三线接口与 SPI、QSPI 以及串行标准相兼容，硬件连接如图 4。12 所示。

SPI 和 AT89C52 的接口传送 8 位字节形式的数据。因此，要把数据输入到 DAC 需要两个写周期。QSPI 接口具有从 8 位至 16 位的可变输入数据长度，可以在一个写周期之内装载 DAC 输入寄存器。

为了更好地使用 TLC5615，建议使用分离的模拟和数据地平面来提高系统性能。设计两个地平面时，应当在低阻抗处将模拟地与数字地连接在一起。通过把器件的 AGND 端连接到系统模拟地平面（该平面能确保模拟地电流流动良好且地平面上的电压降可以忽略），可以实现最佳的接地连接。

VCC 和 AGND 之间应连接一个 0.1μF 的陶瓷旁路电容且应当用短引线安装在尽可能靠近器件的地方。

当系统不使用 D/A 转换器时，把 DAC 寄存器设置为全 0，可以使基准电阻阵列和输出负载的功耗降为最小。

```
;***********************************************************
描述:
使用电压表测量 D/A 电压输出端可以看到最大，中间，最小的电压值 不断跳变
;***********************************************************
  DIN    BIT     P1.0
  SCLK    BIT     P1.1
```

```
    CS     BIT     P1.4
DATA_H    EQU      30H
DATA_L    EQU      31H

          ORG 0000H
          JMP START
          ORG 0030H

START:    MOV P1,#60H              ;禁用其他芯片
    MOV R0,#03FH              ;计数器
    MOV R1,#04H
          MOV DATA_H,#0fH          ;最大值
          MOV DATA_L,#0fcH
          ACALL TCL5615
          ACALL DELAY1
          MOV DATA_H,#07H          ;中间值
          MOV DATA_L,#0fcH
          ACALL TCL5615
    ACALL DELAY1
    MOV DATA_H,#00H           ;最小值
          MOV DATA_L,#00H
          ACALL TCL5615
```

```
        ACALL DELAY1
        JMP START               ;返回
;**************************************************
; 将数据写到 5615
;**************************************************
TCL5615: CLR CS
        ACALL DELAY
        MOV R6,#08H
LOOPH:  LCALL DELAY
        MOV A,DATA_H
        RLC A
        MOV DIN,C
        SETB SCLK
        MOV DATA_H,A
        LCALL DELAY
        CLR SCLK
        DJNZ R6,LOOPH
        MOV R6,#08H
LOOPL:  MOV A,DATA_L
        RLC A
        MOV DIN,C
        SETB SCLK
```

```
            MOV DATA_L,A
            LCALL DELAY
            CLR SCLK
            DJNZ R6,LOOPL
            SETB CS
            RET
DELAY:      MOV R5,#01H
 H0:        MOV R7,#20H
 H1:        DJNZ R7,H1
            DJNZ R5,H0
            RET
DELAY1:     MOV R5,#20H
HH0:        MOV R6,#0ffH
HH1 :       MOV R7,#0ffH
HH2:        DJNZ R7,HH2
            DJNZ R6,HH1
            DJNZ R5,HH0
            RET
            END
```

3－8 带 **RAM** 实时时钟芯片 **DS1302** 及应用程序

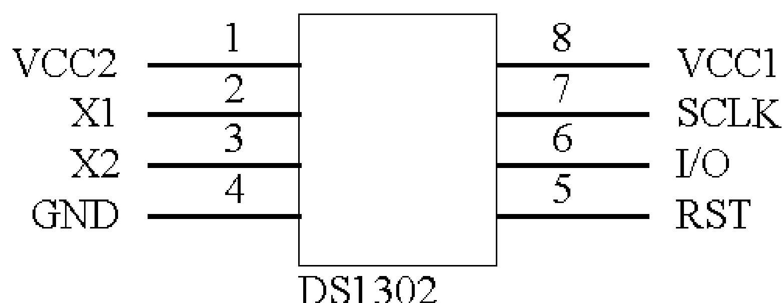
1．1 硬件描述

DS1302 是高性能低功耗时钟芯片,包括实时时钟/日历和 31 字节的静态 RAM.可实时地对秒、分、时、日、周、月以及闰年进行计数处理。内部有 31 字节的高速 RAM，可通过外部可充电池加电长期保存数据，并能慢速为电池充电。通过简单的三线串行方式接口，能在 2.5—5.5V 电源下可靠工作，在 2.5V 时耗电小于 300nA。在主电源关闭的情况下，能保持时钟的连续运行。

DS1302 可广泛应用于智能仪器、单片机系统和家用时钟电路等领域。

该器件外形有八引脚 DIP 及表贴八引脚 SOIC 封装，可选的工业温度范围为-40℃—+85℃。

1、 引脚排列



2、

DS1302 的引脚排列如图所示。脚 X1、X2 是 32 768HZ 晶振输入/输出端，通常要接补偿电容；RST 是复位输入端；I/O 是数据输入/输出端；SCLK 是串行时钟输入端；VCC2 是主电源，一般接+5V；VCC1 是附助电源，一般接 3.6V 可充电池。

3、 使用说明

DS1302 经过一个简单的串行接口与单片机通信。实时时钟/日历

提供秒、分、时、日、周、月和年等信息。对于小于 31 天的月，月末的日期自动进行调整，并包括了闰年校正的功能。时钟的运行可以采用 24 小时或带 AM（上午）/PM（下午）的 12 小时格式。使用同步串行通信。与时钟/RAM 通信仅需三根线：RST（复位)、I/O（数据线）和 SCLK（串行时钟)。数据可以以每次一个字节或多达 31 字节的多字节的形式传送至时钟/RAM 或从其中送出。DS1302 被设计成能在非常低的功耗下工作，消耗小于 1μW 的功率便能保存数据和时钟信息。

DS1302 是 DS1202 的升级产品，除了具有 DS1202 基本的慢速充电功能外，DS1302 还有另外的特点，包括：用于主电源和备分电源的双电源引脚、可编程的 VCC 慢速充电以及有七个附加字节的高速暂存存储器。

1) 命令字节

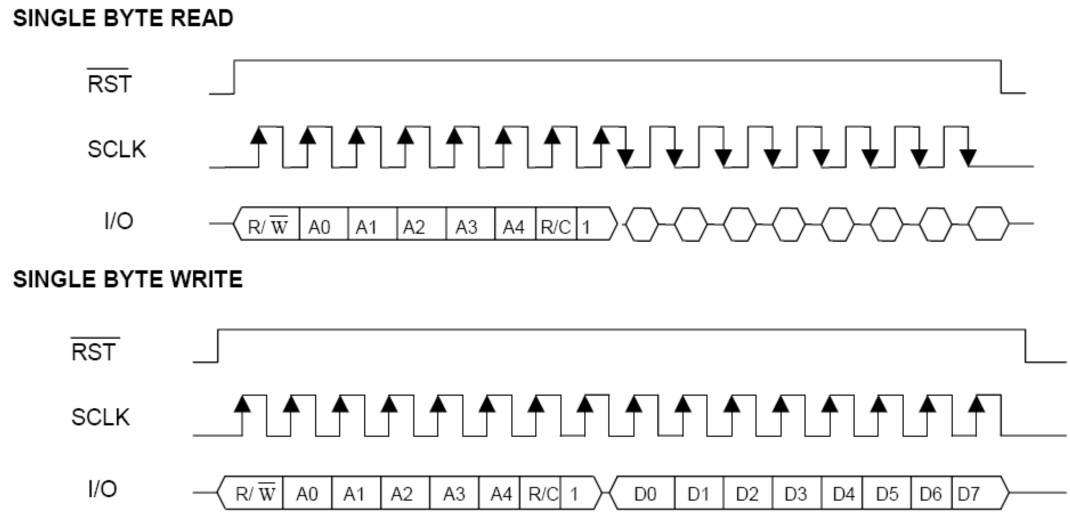
每一数据传送由命令字节初始化。最高有效位 MSB（位 7）必须为逻辑 1。如果它是 0，禁止写 DS1302。位 6 为逻辑 0（CLK)，指定时钟/日历数据；为逻辑 1，指定 RAM 数据。位 1－位 5（A0－A4 地址）指定进行输入或输出的特定寄存器。最低有效位 LSB（位 0）为逻辑 0，指定进行写操作（输入)；为逻辑 1，指定进行读操作（输出)。命令字节总是从最低有效 LSB（位 0）开始输入。命令字节的格式如下：

MSB	6	5	4	3	2	1	LSB
1	RAM/CLK	A4	A3	A2	A1	A0	RD/WR

2) 复位和时钟控制

通过把 RST 输入驱动至高电平来启动所有的数据传送。RST 输入有两种功能。首先，RST 接通控制逻辑，允许地址/命令序列送入移位寄存器。其次，RST 提供了中止单字节或多字节数据传送的手段。

数据输入时，在时钟的上升沿数据必须有效，而数据位在时钟的下降沿输出。如果 RST 输入为低电平，那么所有的数据传送中止且 I/O 引脚变为高阻抗状态。上电时，在 VCC2 ≥2.5V 之前，RST 必须为逻辑 0。此外，当把 RST 驱动至逻辑 1 的状态时，SCLK 必须为逻辑 0。



3) 数据输入与输出

跟随在输入写命令字节的八个 SCLK 周期之后，在下八个 SCLK 周期的上升沿输入数据字节。如果有额外的 SCLK 周期，它们将被忽略。数据从位 0 开始输入。

跟随在输入读命令字节的八个 SCLK 周期之后，在下八个 SCLK 周期的下降沿输出数据字节。注意，被传送的第一个数据位发生在写命令字节的最后一位之后的第一个下降沿。只要 RST 保持为高电平，如果有额外的 SCLK 周期，它们将重新发送数据字节。这一操作使之具

有连续的多字节方式的读能力。另外，在 SCLK 的第一上升沿，I/O 引脚为三态。数据从位 0 开始输出。

4)　　多字节方式

通过对地址 31（十进制）寻址，可以把时钟/日历或 RAM 寄存器规定为多字节方式，如前所述，位 6 规定时钟或 RAM，而位 0 规定读或写，在时钟/日历寄存器中的地址 9 至 31 或 RAM 寄存器中的地址 31 不能存储数据。在多字节方式中，读或写从地址 0 的位 0 开始。

当以多字节方式写时钟寄存器时，必须按数据传送的次序写最先八个寄存器。但是，当以多字节方式写 RAM 时，为了传送数据，不必写所有 31 个字节。不管是否写了全部 31 个字节，所写的每一个字节都将送至 RAM。

5)　　DS1302 内部寄存器

DS1302 内部寄存器地址（命令）及数据寄存器分配情况如图。

REGISTER ADDRESS

A. CLOCK

	7	6	5	4	3	2	1	0
SEC	1	0	0	0	0	0	0	RD/$\overline{W}$
MIN	1	0	0	0	0	0	1	RD/$\overline{W}$
HR	1	0	0	0	0	1	0	RD/$\overline{W}$
DATE	1	0	0	0	0	1	1	RD/$\overline{W}$
MONTH	1	0	0	0	1	0	0	RD/$\overline{W}$
DAY	1	0	0	0	1	0	1	RD/$\overline{W}$
YEAR	1	0	0	0	1	1	0	RD/$\overline{W}$
CONTROL	1	0	0	0	1	1	1	RD/$\overline{W}$
TRICKLE CHARGER	1	0	0	1	0	0	0	RD/$\overline{W}$
CLOCK BURST	1	0	1	1	1	1	1	RD/$\overline{W}$

B. RAM

	7	6	5	4	3	2	1	0
RAM 0	1	1	0	0	0	0	0	RD/$\overline{W}$
⋮								
RAM 30	1	1	1	1	1	1	0	RD/$\overline{W}$
RAM BURST	1	1	1	1	1	1	1	RD/$\overline{W}$

REGISTER DEFINITION

	7	6	5	4	3	2	1	0
00–59	CH	10 SEC			SEC			
00–59	0	10 MIN			MIN			
01–12 00–23	12/24	0	10 A/P	HR	HR			
01–28/29 01–30 01–31	0	0	10 DATE		DATE			
01–12	0	0	0	10 M	MONTH			
01–07	0	0	0	0	0	DAY		
00–99	10 YEAR				YEAR			
	WP	0	0	0	0	0	0	0
	TCS	TCS	TCS	TCS	DS	DS	RS	RS
RAM DATA 0								
⋮								
RAM DATA 30								

时钟/日历与时钟暂停

时钟/日历包含在七个写/读寄存器中。包含在时钟/日历寄存器的数据是二十进制（BCD）码。

秒寄存器的位 7 定义为时钟暂停位。当此位（CH）设置为逻辑 1 时，时钟振荡器停止，DS1302 被置入低功率的备份方式，其电源消耗小

于 100nA；当把此位置逻辑 0 时，时钟将启动。

（1） AM-PM/12-24 方式

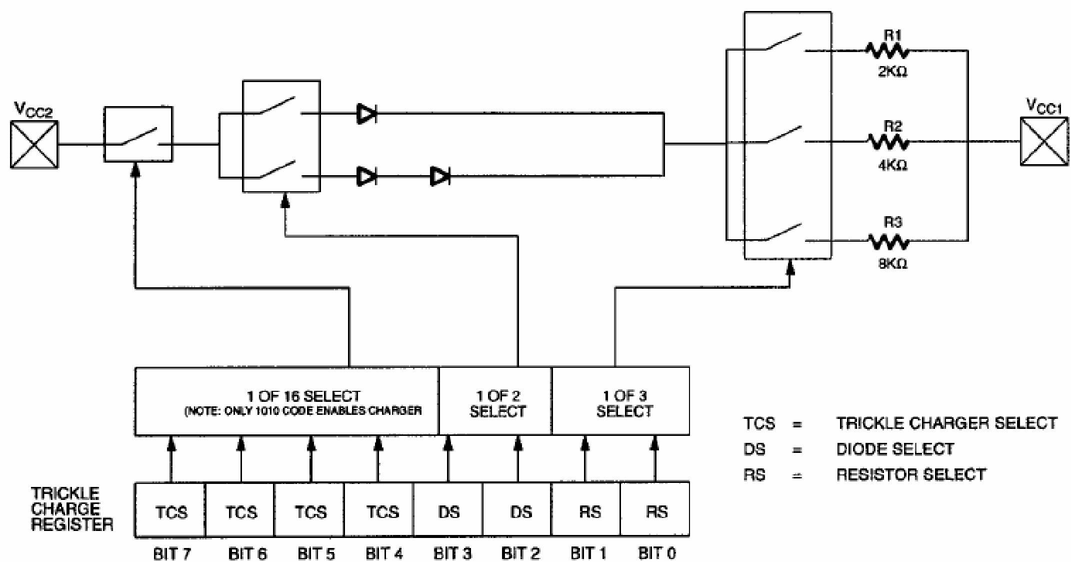
小时寄存器的位 7 定义为 12 或 24 小时方式选择位。当它为高电平时，选择 12 小时方式。在 12 小时方式下，位 5 是 AM/PM 位，此位为逻辑高电平时表示 PM。在 24 小时方式下，位 5 是第二个 10 小时位（20—23 时）。

（2） 写保护寄存器。

写保护寄存器的位 7 是写保护的。开始七位（位 0—6）置为 0，在读操作时总是读出 0。在对时钟或 RAM 进行写操作之前，位 7 必须为 0。当它为高电平时，写保护防止对任何其它寄存器进行写操作。

（3） 慢速充电寄存器

这个寄存器控制 DS1302 的慢速充电特征。图是简化的慢速充电基本电路。



慢速充电选择（TCS）位（位 4—位 7）控制慢速充电器的选择。为了

防止偶然的因素使之工作，只有 1010 模式才能使慢速充电器工作，所有其它模式将禁止慢速充电器。DS1302 上电时，慢速充电器被禁止。二极管选择（DS）位（位 2、位 3）选择是一个二极管还是两个二级管连接在 VCC1 与 VCC2 之间。如果 DS 为 01，那么选择一个二极管；如果 DS 为 10，则选择两个二极管。如果 DS 为 00 或 11，那么充电器被禁止，与 TCS 无关。RS 位（位 0、位 1）选择连接在 VCC1 与 VCC2 之间的电阻。

（4）时钟/日历多字节方式

时钟/日历命令字节可规定多字节方式。在此方式下，最先八个时钟/日历寄存器可以从地址 0 的第 0 位开始连接地读或写。

当指定写时钟/日历为多字节方式时，如果写保护位被设置为高电平，那么没有数据会传送到八个时钟/日历寄存器（包括控制寄存器）的任一个。在多字节方式下，慢速充电器是不可访问的。

（5）单字节与多字节 RAM 方式。

静态 RAM 中 RAM 地址空间可按地址读/写的 31×8 字节。也可设置成多字节工作方式。在此方式下，可以从地址 0 的第 0 位开始顺序读或写 31 个字节的 RAM 寄存器。

（6）晶振的选择

32 768HZ 的晶振可通过引脚 2 和 3（X1、X2）直接连接至 DS1302。所选用晶振规定的负载电容量（CL）应当为 6pF。

然而，许多人在选用晶振时仅仅注意了晶振的额定频率值，而忽视了晶振的负载电容大小，甚至连许多经销商也不能提供所销晶振的负载

电容。所以，即使在使用中选用了符合 32 768HZ 的晶振，但如果该晶振的负载电容与 DS1302 提供的 6pF 不一致时，就会影响晶振的起振或导致振荡频率的偏移。一般可通过下述方法解决：

①当所选的晶振负载电容不是 6pF 时，可以采用增加辅助电容的方法提高或降低 DS1302 振荡器的电容负载，使之与晶体所需的电容值匹配。

如果已知晶体的负载电容为 C1，若 C1<6pF，则可以增加一个并联电容 CS，以产生所需的总负载电容为 C1，即 C1=6pF+CS；若 C1>6pF，则可以在晶体的一端增加一个串联电容 CS，以产生所需的负载电容 C1，即 1/C1=1/6pF+1/CS，通过计算即可得出应增加的辅助电容大小。

②在使用前对晶体的负载电容并不知道其值的情况下，通过测定晶体振荡频率的方法可以确定该晶体负载电容的值。

对于晶体振荡器来说，其振荡频率与负载电容之间的关系是确定的。以 DS1302 使用的 32 768HZ 晶振为例：当它工作于所要求的负载电容时，能较准确地产生 32 768HZ 的频率；当它的负载电容小于 6pF 时，其振荡频率会正向偏移；当它的负载电容大于 6pF 时，其振荡频率就会负向偏移。因此，对于未知负载电容的晶体，应首先采用实验的方法，在其两端加入辅助电容使晶体起振，然后由频率计测出振荡频率。若测得频率大于 32 768HZ，说明负载电容偏小；若测得频率小于 32 768HZ，说明负载电容偏大。对辅助电容逐步调整，最终使振荡频率尽可能接近 32 768HZ，则此时晶体端所接负载电容的总和就是适合该晶体的负载电容。

(7) 电源控制

VCC1 是为 DS1302 提供的备用电源，一般可接 3V 干电池或 3.6V 蓄电池（以便充电）。

VCC2 是为 DS1302 提供的主电源。在这种运用方式中，VCC2 给芯片供电，并通过内部为备用电源充电，以便在没有主电源的情况下能保存时间信息及数据。

DS1302 由 VCC1 或 VCC2 两者中较大者供电。当 VCC2 大于 VCC1+0.2V 时，VCC2 给 DS1302 供电；当 VCC2 小于 VCC1 时，DS1302 由 VCC1 供电。

```
;**********************************
;文件名:LED1302.asm
;功能   :在 LJD-SY2A 实验板完成设置 DS1302 时间为 08 年 8 月 8 日
19 点 59 分 59 秒，
;显示时、分、秒。
;**********************************
LOAD        BIT      P1.2
DIN         BIT      P1.0
DCLK        BIT      P1.1
DOUT        BIT      P3.2
BUZZER      BIT      P2.6
;**********************************
IO_DATA     BIT      P1.0
```

```
SCLK        BIT      P1.1
RST         BIT      P1.3
;**********************************

BitCnt      DATA     30H        ; 数据位计数器
ByteCnt     DATA     31H        ; 数据字节计数器
Command     DATA     32H        ; 命令字节地址
RcvDat      DATA     40H        ; 接收数据缓冲区
XmtDat      DATA     50H        ; 发送数据缓冲区
;*****************************
USERFLAG    EQU      20H
BCD4A       EQU      22H
BCD4B       EQU      23H

;**********************************
        ORG      0000H
        JMP      START

        ORG      0030H
START:  MOV  P1,#60H
        MOV      SP,#7FH
;       CLR     BUZZER
```

```
;**********************************
          CLR       DIN             ;初始化 CH451
          SETB      DCLK
          SETB      DIN
          SETB      LOAD
          SETB      DOUT
          NOP
          MOV       B,#04H          ;设置 CH451
          MOV       A,#03H          ;关看门狗开显示键盘
         LCALL     WRITE
         NOP
;********************设置时钟****************************
Write_Enable:
               MOV Command,#8Eh          ;命令字节为 8E
               MOV ByteCnt,#1            ;单字节传送模式
               MOV R0,#XmtDat            ;数据地址覆给 R0
               MOV XmtDat,#00h           ;数据内容为 0 写入允许
               LCALL Send_Byte           ;调用写入数据子程序
;******************************当把秒寄存器的第 7 位时钟停
止位设置为 0 时起动时钟开始
              MOV Command,#80h            ; 命令字节为 80
              MOV ByteCnt,#1               ; 单字节传送模式
```

```
        MOV R0,#XmtDat                  ;数据地址覆给 R0
        MOV XmtDat,#00h                 ;数据内容为 0 振荡器
工作允许
        LCALL Send_Byte                 ;调用写入数据子程序

;*******************************************
Write_Multiplebyte:
        MOV Command,#0BEh               ;命令字节为 BEh
        MOV ByteCnt,#8                  ;多字节写入模式此模
块为 8 个
        MOV R0,#XmtDat                  ;数据地址覆给 R0
        MOV XmtDat,  #59h               ;秒单元内容为 59h
        MOV XmtDat+1,#59h               ;分单元内容为 59h
        MOV XmtDat+2,#19h               ;时单元内容为 13h
        MOV XmtDat+3,#08h               ;日期单元内容为 21h
        MOV XmtDat+4,#08h               ;月单元内容为 06h
        MOV XmtDat+5,#05h               ;星期单元内容为 03h
        MOV XmtDat+6,#08h               ;年单元内容为 00h
        MOV XmtDat+7,#0                ;写保护单元内容为 00h
        LCALL Send_Byte                 ;调用写入数据子程序
;**********************************************
Read_A1:
```

```
        MOV     B,#05H                  ;设置为译码
        MOV     A,#08AH
        LCALL   WRITE
;*************************************************
TIME:
          MOV     Command,#85h              ; 命令字节为 85h
          MOV     ByteCnt,#1                ; 单字节传送模式
          MOV     R1,#RcvDat                ;数据地址覆给 R1
          LCALL   Receive_Byte               ;调用读出数据子
程序
          LCALL   BCD8_1
          MOV       B,#0aH
          MOV     A,BCD4A
          LCALL   WRITE
          ORL     BCD4B,#80H
          MOV     B,#0bH
          MOV     A,BCD4B
          LCALL   WRITE
;**********************************************
          MOV     Command,#83h
          MOV     ByteCnt,#1
          MOV     R1,#RcvDat
```

```
        LCALL   Receive_Byte
    LCALL   BCD8_1
        MOV      B,#0cH
      MOV      A,BCD4A
        LCALL   WRITE
        ORL     BCD4B,#80H
        MOV     B,#0dH
      MOV      A,BCD4B
        LCALL   WRITE
;**********************************************
        MOV     Command,#81h
        MOV     ByteCnt,#1
        MOV     R1,#RcvDat
        LCALL   Receive_Byte
        LCALL   BCD8_1
        MOV      B,#0eH
        MOV     A,BCD4A
        LCALL   WRITE
        MOV     B,#0fH
        MOV     A,BCD4B
        LCALL   WRITE
        NOP
```

```
            LJMP     TIME
;**********************************
;写 CH451 程序
;文件名：WRITE
;入参：B、ACC 待写的 12 位数据，低 8 位在 ACC 中，高 4 位在 B 的
低四位中。
;**********************************
WRITE:
       ;      PUSH      DPH
        ;     PUSH      DPL
        ;     PUSH      ACC
              CLR        EX0
       CLR         LOAD
       MOV         R7,#08H
WRITE_1:
          RRC          A
       CLR         DCLK
       MOV         DIN,C
       SETB        DCLK
       DJNZ        R7,WRITE_1
       MOV         A,B
       MOV         R7,#004H
```

```
WRITE_2:
            RRC         A
        CLR         DCLK
        MOV         DIN,C
        SETB        DCLK
        DJNZ        R7,WRITE_2
        SETB        LOAD
        SETB        EX0
    ;    POP         ACC
    ;    POP         DPH
     ;   POP         DPL
        RET
;*********************************
BCD8_1:
            MOV         R7,#04H
            MOV         A,RcvDat
RR1:
            CLR         C
            RRC         A
            DJNZ        R7,RR1
            MOV         BCD4A,A
            NOP
```

```
            MOV      R7,#04H
            MOV      A,RcvDat
RL1:
        CLR      C
        RLC      A
        DJNZ     R7,RL1
        SWAP     A
        MOV      BCD4B,A
        RET
;*************************************
Send_Byte:
            CLR     RST
            NOP
            CLR     SCLK
            NOP
            SETB    RST
            NOP
            MOV     A,Command
            MOV     BitCnt,#08h
S_Byte0:
            RRC     A
            MOV     IO_DATA,C
```

```
        NOP
        SETB    SCLK
        NOP
        CLR     SCLK
        DJNZ    BitCnt,S_Byte0
        NOP
S_Byte1:
        MOV     A,@R0
        MOV     BitCnt,#08h
S_Byte2:
        RRC     A
        MOV     IO_DATA,C
        NOP
        SETB    SCLK
        NOP
        CLR     SCLK
        DJNZ    BitCnt,S_Byte2
        INC     R0
        DJNZ    ByteCnt,S_Byte1
        NOP
        CLR     RST
        RET
```

```
;************************************************************
******************************
;接收数据程序;
;名称:Receive_Byte
;描述:从被控器 DS1302 接收 ByteCnt 个字节数据
;命令字节地址在 Command 中
;所接收数据的字节数在 ByteCnt 中接收的数据在 RcvDat 缓冲区中
;************************************************************
*************************
Receive_Byte:
             CLR     RST                   ;
             NOP
             CLR     SCLK
             NOP
             SETB     RST
             MOV      A,Command

             MOV      BitCnt,#08h
R_Byte0:
             RRC      A
             MOV      IO_DATA,C
             NOP
```

```
        SETB    SCLK
        NOP
        CLR     SCLK
        DJNZ    BitCnt,R_Byte0
        NOP
R_Byte1:
        CLR     A
        CLR     C
        MOV     BitCnt,#08h
R_Byte2:
        NOP
        MOV     C,IO_DATA
        RRC     A
        SETB    SCLK
        NOP
        CLR     SCLK
        DJNZ    BitCnt,R_Byte2
        MOV     @R1,A
        INC     R1
        DJNZ    ByteCnt,R_Byte1
        NOP
        CLR     RST
```

```
            RET
;************************************
END
```

3－9 用单线数字温度传感器 **DS18B20** 实现温度测量

来自控制现场的诸如温度、压力等连续变化的物理量经传感器和变送单元转换后成为一定形式的模拟量。为了将这些量送入单片机，必须进行 A/D 转换，将模拟量变成数字量。

1．1 DS18B20 概述

DS18B20 是 Dallas 公司推出的单线数字式测温芯片，它能在现场采集温度数据，并将温度数据直接转换成数字量输出。

1． DS18B20 的内部结构、引脚和工作说明

图 14-6 是 DS18B20 的内部结构图。

DS18B20 只有 3 个引脚，说明如下：

DQ-数据输入输出。漏极开路 1 线接口。也在寄生电源模式时给设备提供电源。

VDD-可选的电源电压脚。VDD 在寄生电源模式时必须接地。

GND-地

DS18B20 的 64 位 ROM 保存了设备的惟一序列码。高速闪存（scratchpad）包含 2 字节的温度寄存器，保存了温度传感器的数字输出。该闪存还提供了对上限（TH）和下限（TL）超标报警寄存器、

配置寄存器（各 1 字节）的访问。TH、TL 和配置寄存器是非易失性的（EEPROM），系统掉电时它们会保存数据。

DS18B20 利用 Dallas 的单总线控制协议，实现了利用单线控制信号在总线上进行通信。由于所有的设备通过漏极开路端(即 DS18B20 的 DQ 脚）连在总线上，控制线需要一个上拉电阻（大约 5K)。在这一总线系统中，微控制器（主控设备）通过惟一的 64 位序列码识别和访问总线上的器件。由于每一设备有惟一的编码，连在一条总线上可被访问的器件数实际上是无限的。

DS18B20 的另一个特点是在没有外部电源下操作的能力。电源由总线为高电平时 DQ 脚上的上拉电阻提供（寄生供电模式)，此时 VDD 脚接地。

另外，DS18B20 也可用传统方式供电，此时将外部电源连在 VDD 脚上即可。

2． DS18B20 的寄存器

DS18B20 存储器组织如图 14.7 所示。字节 0 和字节 1 分别包含温度寄存器的 LSB 和 MSB，这些字节是只读的。字节 2 和 3 提供了对 TH（上限报警触发寄存器）和 TL（下限报警触发寄存器）的访问。字节 4 包含配置寄存器数据。字节 5、6 和 7 保留作器件内部使用，不能被改写；当读时，这些字节返回全 1 值。字节 8 是只读的，含有字节 0 到字节 7 的 CRC 校验。

高速闪存的第 4 个字节包含配置寄存器，其组织结构如图 14-8 所示。用户可以用这一寄存器的 R0 和 R1 位设置 DS18B20 的转换分辨

率（见表 14-1）。这些位上电后默认值是 R0=1 和 R1=1（12 位分辨率）。第 7 位和 0~4 位保留作内部使用，不能更改；在读时这些位返回 1。

3． DS18B20 的读写

访问 DS18B20 的顺序如理

*初始化；

*ROM 命令（接着是任何需要的数据交换）；

*DS18B20 函数命令（接着是任何需要的数据交换）。

每一次访问 DS18B20 时必须遵循这一顺序，如果其中的任何一步缺少或打乱它们的顺序，DS18B20 将不会响应。

（1）初始化时序

所有与 DS18B20 的通信首先必须初始化：控制器发出复位脉冲，DS18B20 以存在脉冲响应。在图 14-9 中给出了描述。当 DS18B20 发出存在脉冲对复位响应时，它指示控制器该 DS18B20 已经在总线上并准备好操作。

（2）读/写时序

控制器在写时序写数据到 DS18B20，在读时序从 DS18B20 中读数据。每一总线时序传送 1 位数据。

读/写时序图见图 14-10。

（3）写流程时序

有两种类型的写时序：写 1 时序和写 0 时序。控制器用写“1”时序写逻辑“1”到 DS18B20，用写“0”时序写逻辑“0”到 DS18B20。所有写时序必须持续最少 60μs，每个写时序之间必须有至少 1μs

的恢复时间。两种类型的写时序都从控制器把总线拉低开始。

为产生写“1”时序，在将总线拉低后，总线控制器必须在 15μs 内释放总线。总线释放后，5K 的上拉电阻将总线电平抬高。为产生写“0”时序，在将总线拉低后，控制器在整个时序内必须持续控制总线为低电平（至少 60μs)。

DS18B20 在控制器发出写时序后的 15~60μs 的时间窗口内采样总线。如果在采样窗口期间总线为高,“1”就被写入 DS18B20；如果在采样窗口期间总线为低,“0”就被写入 DS18B20。

（4）读时序

当控制器发出读时序时，DS18B20 可以发送数据到控制器。

所有读时序必须持续最少 60μs，每个读时序之间必须有至少 1μs 的恢复时间。读时序从控制设备将总线拉低至少 1μs 后释放总线开始。控制器启动读时序后，DS18B20 开始在总线上传送“1”或“0”时，DS18B20 通过保持总线为高发送“1”，将总线拉低发送“0”发送“0”时，DS18B20 在时序结束时释放总线，总线被上拉电阻拉回到高电平空闲状态。　　　从 DS18B20 输出的数据在启动读时序的下降沿后 15μs 内有效。因此，控制器必须在时序开始的 15μs 内释放总线然后采样总线状态。

通过读写时序，控制器可以发出控制命令，对 DS18B20 进行读写操作。

4.　　DS18B20 常用命令

（1）SKIP ROM [CCH]

控制器可以用这一命令同时访问总线上的所有设备而不需要送出 ROM 序列码信息。例如：发出 Skip ROM 命令后接着送出 Convert T［44H］命令，控制器可以使总线上的所有 DS18B20 同时进行温度转换。请注意，仅当总线上只有一个从属设备时，Skip ROM 命令后才可跟着 Read Scratchpad［BEH］命令。如果总线上有多于一个的从属设备，由于多设备企图同时送出数据，将引起数据冲突。

（2）SEARCH ROM［F0H］

当系统开始上电时，控制器必须识别总线上所有从机的 ROM 序列码，以确定从机的数目和它们的类别。控制器需要执行 Search ROM 循环（如 Search ROM 命令后接着数据交换）足够多次以识别所有的从属设备。如果只有一个从属设备在总线上，可使用简单的 Read ROM 命令取代 Search ROM。每一个 Search ROM 命令之后必须返回到事务序列的步骤 1（初始化）。

（3）READ ROM［33H］

这一命令在总线上只有一个从属设备时使用。它使得控制器可以不用 Search ROM 命令就可读出从机的 64 位 ROM 序列码。当多于一个从属设备在总线上时，如果使用该命令，由于所有设备都企图响应该命令将产生数据冲突。

（4）CONVERT T［44H］

这一命令开始一次温度转换。变换结束后，数据保存在暂存器的 2 字节温度寄存器中，DS18B20 回到低功耗空闲状态。如果设备工作在寄生电源模式，则这一命令发出后 10μs 之内，在整个变换期间

（tconv）控制器必须在总线上能够有强的上拉。如果 DS18B20 由外部电源供电，那么 Convert T 命令之后控制器可以发出读时序。如果温度变换正在进行，那么 DS18B20 返回“ 0 ”; 如果已经完毕，则返回“ 1 ”。在寄生供电模式下，不能使用这一技术，因为在变换期间总线被抬高。

（5） WRITE SCRATCHPAD [4EH]

这一命令使得控制器可以写 3 字节数据到 DS18B20 的暂存器中。第一字节数据写到 TH 寄存器（暂存器的字节 2），第 2 字节写到 TL 寄存器（字节 3），第 3 字节写到配置寄存器（字节 4）。数据以最低有效位先发送。所有 3 字节必须在控制器发出复位或数据可能丢失之前写完。

（6） READ SCRTCHPAD [BEH]

这一命令使控制器可以读暂存器的内容。数据传送开始于字节 0 的最低位，直到暂存器的第 9 字节（字节 8CRC）被读取。任何时候，如果只需部分暂存器数据，控制器可以使用复位结束读操作。

温度传感器 ds1820 的汇编程序

```
TEMPER_L     EQU     36H
TEMPER_H     EQU     35H
TEMPER_NUM   EQU     60H
FLAG1        BIT     00H
DQ           BIT     P3.3
AAA:    MOV    SP,#70H
```

```
        LCALL GET_TEMPER
        LCALL TEMPER_COV
        LJMP  AAA
        NOP
        ;-----------------读出转换后的温度值
GET_TEMPER:
        SETB  DQ              ; 定时入口
BCD:    LCALL INIT_1820
        JB    FLAG1,S22
        LJMP  BCD          ; 若 DS18B20 不存在则返回
S22:    LCALL DELAY1
        MOV   A,#0CCH         ; 跳过 ROM 匹配------0CC
        LCALL WRITE_1820
        MOV   A,#44H          ; 发出温度转换命令
        LCALL WRITE_1820
        NOP
        LCALL DELAY
        LCALL DELAY
CBA:    LCALL INIT_1820
        JB    FLAG1,ABC
        LJMP  CBA
ABC:    LCALL DELAY1
```

```
        MOV    A,#0CCH          ; 跳过 ROM 匹配
        LCALL WRITE_1820
        MOV    A,#0BEH          ; 发出读温度命令
        LCALL WRITE_1820
        LCALL READ_18200     ;READ_1820
        RET
         ; 读 DS18B20 的程序,从 DS18B20 中读出一个字节的数据
READ_1820:
        MOV     R2,#8
RE1:
        CLR     C
        SETB     DQ
        NOP
        NOP
        CLR     DQ
        NOP
        NOP
        NOP
        SETB     DQ
        MOV     R3,#7
        DJNZ     R3,$
        MOV     C,DQ
```

```
        MOV     R3,#23
        DJNZ     R3,$
        RRC     A
        DJNZ     R2,RE1
        RET
        ;-------------------写 DS18B20 的程序
WRITE_1820:
        MOV     R2,#8
        CLR     C
WR1:
        CLR     DQ
        MOV     R3,#6
        DJNZ    R3,$
        RRC     A
        MOV     DQ,C
        MOV     R3,#23
        DJNZ    R3,$
        SETB    DQ
        NOP
        DJNZ     R2,WR1
        SETB     DQ
        RET
```

;-------------------读 DS18B20 的程序,从 DS18B20 中读出两个字节的温度数据

```
READ_18200:
        MOV     R4,#2               ; 将温度高位和低位从 DS18B20 中读出
        MOV     R1,#36H             ; 低位存入 36H(TEMPER_L),高位存入 35H(TEMPER_H)
RE00:
        MOV     R2,#8
RE01:
        CLR     C
        SETB    DQ
        NOP
        NOP
        CLR     DQ
        NOP
        NOP
        NOP
        SETB    DQ
        MOV     R3,#7
        DJNZ    R3,$
        MOV     C,DQ
```

```
        MOV     R3,#23
        DJNZ    R3,$
        RRC     A
        DJNZ    R2,RE01
        MOV     @R1,A
        DEC     R1
        DJNZ    R4,RE00
        RET
        ;-------------------将从 DS18B20 中读出的温度数据进行转换
TEMPER_COV:
        MOV     A,#0F0H
        ANL     A,TEMPER_L          ; 舍去温度低位中小数点后的四位温度数值
        SWAP    A
        MOV     TEMPER_NUM,A
        MOV     A,TEMPER_L
        JNB     ACC.3,TEMPER_COV1      ; 四舍五入去温度值
        INC     TEMPER_NUM
TEMPER_COV1:
        MOV     A,TEMPER_H
        ANL     A,#07H
```

```
        SWAP    A
        ORL     A,TEMPER_NUM
        MOV     TEMPER_NUM,A     ; 保存变换后的温度数据
        LCALL   BIN_BCD
        RET
      ;-------------------将16进制的温度数据转换成压缩BCD码
BIN_BCD:
        MOV     DPTR,#TEMP_TAB
        MOV     A,TEMPER_NUM
        MOVC     A,@A+DPTR
        MOV     TEMPER_NUM,A
        RET
TEMP_TAB:
        DB     00H,01H,02H,03H,04H,05H,06H,07H
        DB     08H,09H,10H,11H,12H,13H,14H,15H
        DB     16H,17H,18H,19H,20H,21H,22H,23H
        DB     24H,25H,26H,27H,28H,29H,30H,31H
        DB     32H,33H,34H,35H,36H,37H,38H,39H
        DB     40H,41H,42H,43H,44H,45H,46H,47H
        DB     48H,49H,50H,51H,52H,53H,54H,55H
        DB     56H,57H,58H,59H,60H,61H,62H,63H
        DB     64H,65H,66H,67H,68H,69H,70H,71H
```

```
        DB      72H,73H,74H,75H,76H,77H,78H,79H
        DB      80H,81H,82H,83H,84H,85H,86H,87H
        DB      88H,89H,90H,91H,92H,93H,94H,95H
        DB      96H,97H,98H,99H
        ;-------------------DS18B20 初始化程序
INIT_1820:
        SETB    DQ
        NOP
        CLR     DQ
        MOV     R0,#80H
TSR1:
        DJNZ     R0,TSR1         ; 延时
        SETB     DQ
        MOV     R0,#25H          ;96US-25H
TSR2:
        DJNZ     R0,TSR2
        JNB     DQ,TSR3
        LJMP     TSR4             ; 延时
TSR3:
        SETB     FLAG1      ; 置标志位,表示 DS1820 存在
        LJMP     TSR5
TSR4:
```

```
        CLR     FLAG1       ; 清标志位,表示 DS1820 不存在
        LJMP     TSR7
TSR5:
        MOV     R0,#06BH            ;200US
TSR6:
        DJNZ     R0,TSR6            ; 延时
TSR7:
        SETB     DQ
        RET
        ;------------------重新写 DS18B20 暂存存储器设定值
RE_CONFIG:
        JB      FLAG1,RE_CONFIG1      ; 若 DS18B20 存在,转
RE_CONFIG1
        RET
RE_CONFIG1:
        MOV     A,#0CCH             ; 发 SKIP ROM 命令
        LCALL     WRITE_1820
        MOV     A,#4EH            ; 发写暂存存储器命令
        LCALL     WRITE_1820
        MOV     A,#00H            ; TH(报警上限)中写入 00H
        LCALL     WRITE_1820
        MOV     A,#00H            ; TL(报警下限)中写入 00H
```

```
        LCALL     WRITE_1820
        MOV     A,#7FH            ; 选择 12 位温度分辨率
        LCALL     WRITE_1820
        RET
        ;------------------延时子程序
DELAY:  MOV   R7,#00H
MIN:    DJNZ  R7,YS500
        RET
YS500:  LCALL YS500US
        LJMP  MIN
YS500US:MOV   R6,#00H
        DJNZ  R6,$
        RET
DELAY1: MOV   R7,#20H
        DJNZ  R7,$
        RET
```

3－10 串行通讯

可以通过串行通信电缆与计算机进行通讯，实现数据信息的交换。其引脚说明如下：

引脚	名称	功能
2	RXD	接 PC 机的 RXD 信号线
3	TXD	接 PC 机的 TXD 信号线
5	GND	信号地
1、4、6 7、8、9	空	未用

串口接收程序如下：

```
;-----------------------------------------------------------
        ORG     0000H
        JMP     START
START:  MOV     SP,#60H           ; 设定堆栈区
        MOV     SCON,#01010000B ; 设定串列方式 MODE1,接收时 REN=1
        MOV     TMOD,#20H         ; 设定计时器 1 为 模式 2
        ORL     PCON,#10000000B ; SMOD=1
        MOV     TH1,#0F4H         ; 设定波特率为 4800
        SETB    TR1               ; 计时器 1 ,开始计时
        MOV     R0,#30H
        MOV     R7,#05H
AGAIN:
        JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        MOV     @R0,A
        INC     R0
```

```
        DJNZ     R7,AGAIN
         RET
  END
;----------------------------------------------------------
```

串口发送程序如下：

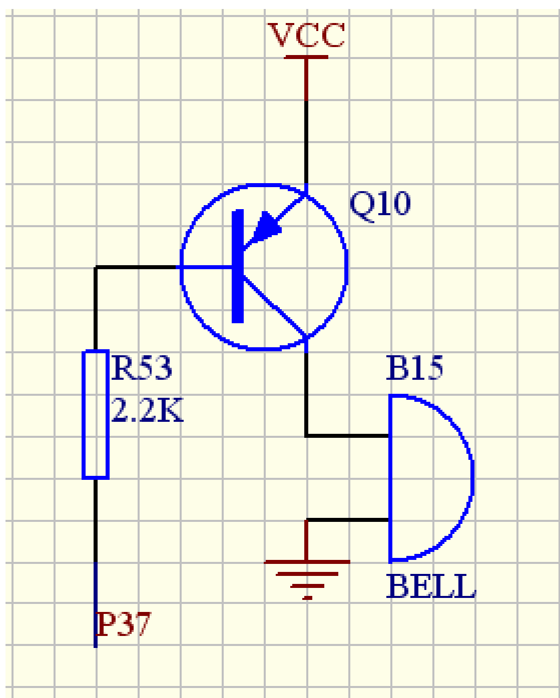
```
;----------------------------------------------------------
         ORG     0000H
         JMP      START
START:  MOV      SP,#60H          ; 设定堆栈区
         MOV      SCON,#01010000B ; 设定串列方式 MODE1,接收
时 REN=1
        MOV      TMOD,#20H        ; 设定计时器 1 为 模式 2
        ORL       PCON,#10000000B ; SMOD=1
        MOV      TH1,#0F4H         ; 设定波特率为 4800
        SETB     TR1               ; 计时器 1 ,开始计时
        MOV       A,#30H              ; 0...9 的 ASCII 码为
30H...39H
AGAIN:
        MOV    SBUF,A            ; 送发送缓冲区
        JNB      TI,$             ; 等待发送完成
        CLR      TI
        INC      A              ; 发送下一个
```

```
        CJNE    A,#3AH, AGAIN
        RET
END
;------------------------------------------------------------
```

3－11 音乐的应用

硬件设计与基本概念

要产生音频脉冲，只要算出某一音频的周期（1/频率），然后将此周期除以 2，即为半周期的时间。利用定时器计时这个半周期时间，每当计时到后就将输出脉冲的 I/O 反相，然后重复计时此半周期时间再对 I/O 反相，就可在 I/O 脚上得到此频率的脉冲。利用单片机内部定时器使其工作在计数器模式 MODE1 下，改变计数值 TH0 及 TL0 以产生不同频率的方法。



电路如图所示，蜂鸣器的动作由 P3.7 控制。当输出低电平时蜂鸣器发出响声，输出高电平时不响，我们只要控制蜂鸣器输出高低电平的时间和变化频率就可以让蜂鸣器发出悦耳的音乐。

实例程序如下：

```
*************************************
LJD-SY-5100\example\Buzz\happybirthday.asm
*************************************
ORG 0000H
JMP START
ORG 000BH
JMP TIM0
START:MOV TMOD,#01H
      MOV IE,#82H
START0:MOV 30H,#00H
NEXT:  MOV A,30H
       MOV DPTR,#TABLE
       MOVC A,@A+DPTR
       MOV R2,A
       JZ END0
       ANL A,#0FH
       MOV R5,A
       MOV A,R2
```

```
        SWAP A
        ANL A,#0FH
        JNZ SING
        CLR TR0
        JMP D1
SING:   DEC A
        MOV 22H,A
        RL  A
        MOV DPTR,#TABLE1
        MOVC A,@A+DPTR
        MOV TH0,A
        MOV 21H,A
        MOV A,22H
        RL  A
        INC A
        MOVC A,@A+DPTR
        MOV TL0,A
        MOV 20H,A
        SETB TR0
D1:     CALL DELAY
        INC  30H
        JMP  NEXT
```

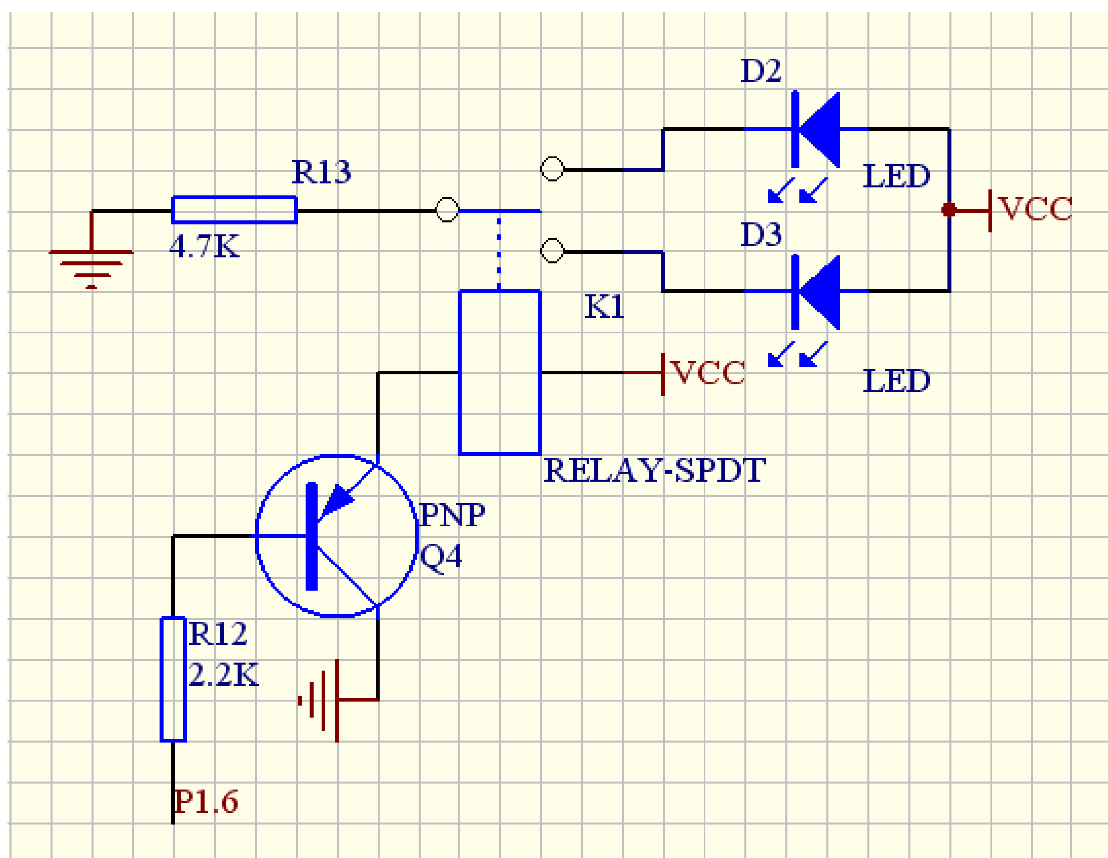
```
ENDO:           CLR  TRO
                JMP  STARTO
TIMO:           PUSH  ACC
                PUSH  PSW
      MOV   THO,21H
      MOV   TLO,20H
      CPL   P3.7
      POP   PSW
      POP   ACC
      RETI
DELAY:      MOV  R7,#02
D2:      MOV  R4,#187
D3:      MOV  R3,#248
      DJNZ R3,$
      DJNZ R4,D3
      DJNZ R7,D2
      DJNZ R5,DELAY
      RET
TABLE1:
   DW 64260,64400,64524,64580
   DW 64684,64777,64820,64898
   DW 64968,65030,65058,65110
```

```
   DW 65157,65178,65217
TABLE:
DB 82H,01H,81H,94H,84H
DB 0B4H,0A4H,04H
DB 82H,01H,81H,94H,84H
DB 0C4H,0B4H,04H
DB 82H,01H,81H,0F4H,0D4H
DB 0B4H,0A4H,94H
DB 0E2H,01H,0E1H,0D4H,0B4H
DB 0C4H,0B4H,04H
DB 82H,01H,81H,94H,84H
DB 0B4H,0A4H,04H
DB 82H,01H,81H,94H,84H
DB 0C4H,0B4H,04H
DB 82H,01H,81H,0F4H,0D4H
DB 0B4H,0A4H,94H
DB 0E2H,01H,0E1H,0D4H,0B4H
DB 0C4H,0B4H,04H
DB 00H
END
```

3－12 继电器

<u>硬件设计与基本概念</u>

继电器广泛用于生产控制的电力系统中，其作用是利用它的常闭和常开触点进行电路切换。由于继电器是利用改变金属触点位置，使动触点和定触点闭合或分开，所以具有接触电阻小，流过电流大和耐压高等优点，特别适用于大电流高电压的使用场合。小型继电器也常用作精密测量电路的转换开关。




```
MAIN:     CLR    p3.6          ;继电器吸合
          setb   p3.7               ;蜂鸣器停叫
          LCALL  DELAY
          SETB   P3.6               ;继电器断开
```

```
            LCALL DELAY
            NOP
            NOP
            CLR p3.7                  ;蜂鸣器鸣叫
            LCALL DELAY
            JMP MAIN
DELAY:      MOV R5,#06H
HH0:        MOV R6,#0FFH
HH1 :       MOV R7,#0ffH
HH2:        DJNZ R7,HH2
            DJNZ R6,HH1
            DJNZ R5,HH0
            RET
           END
```

第四章 下载调试的使用

本实验板自带一片调试监控芯片，使该监控的调试直接借助 Keil C51 软件进行，使开发更加方便快捷。

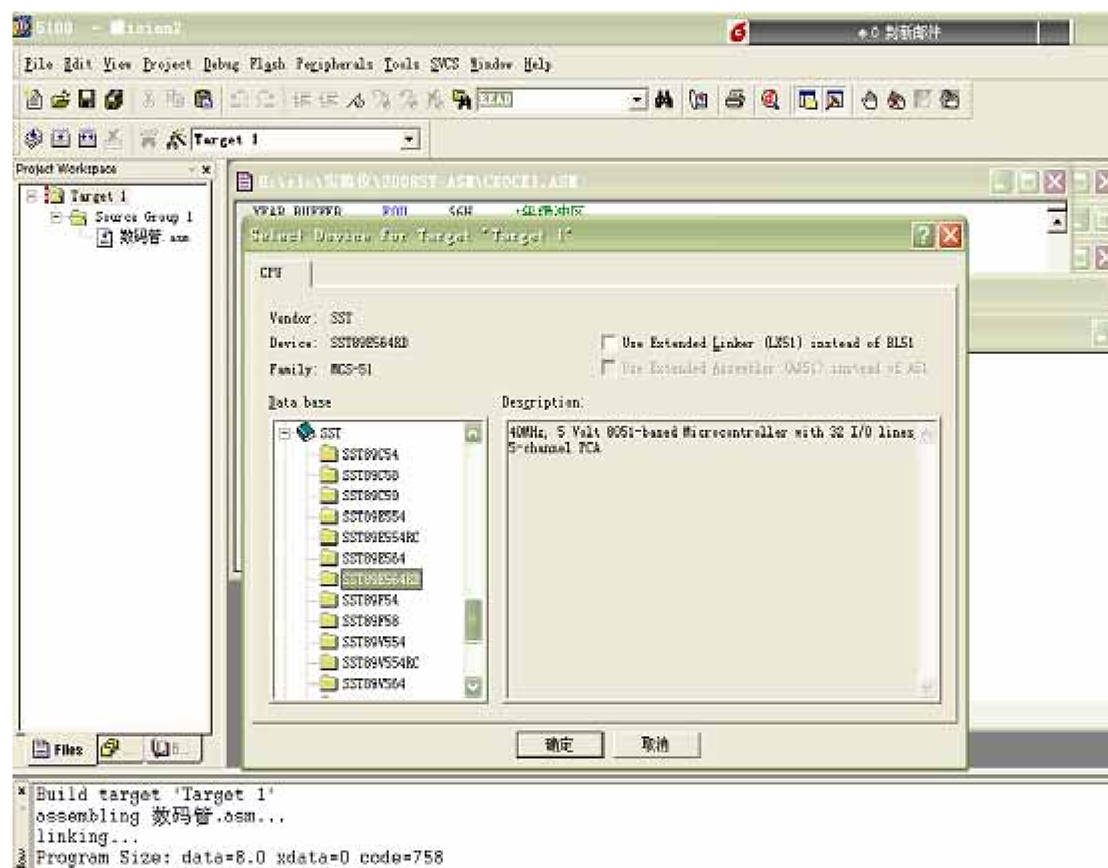
注意:

此调试监控只是为了方便没有仿真器的用户做简单的调试之用，由于版权关系，在用户使用此监控软件过程中出现的调试问题，均不属于本公司以及本产品的责任范围，

请依照下面步骤进行连接即可：

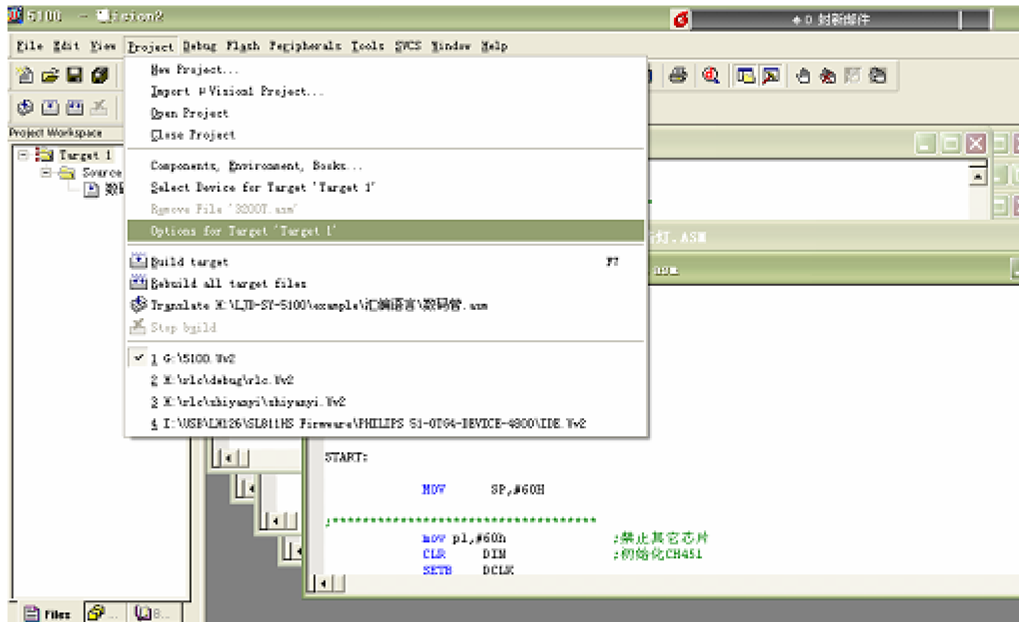
把开发板上的用户 CPU（STC89C52RC）取下，换上调试监控，用标准 RS232 连接线把实验板的串口和计算机的 RS232 口连接好。

步骤一：生成 Project 和编辑用户程序。选择一个 SST MCU（SST89E54D2、SST89E58RD 和 SST89E/V564RD 等）做目标器件。

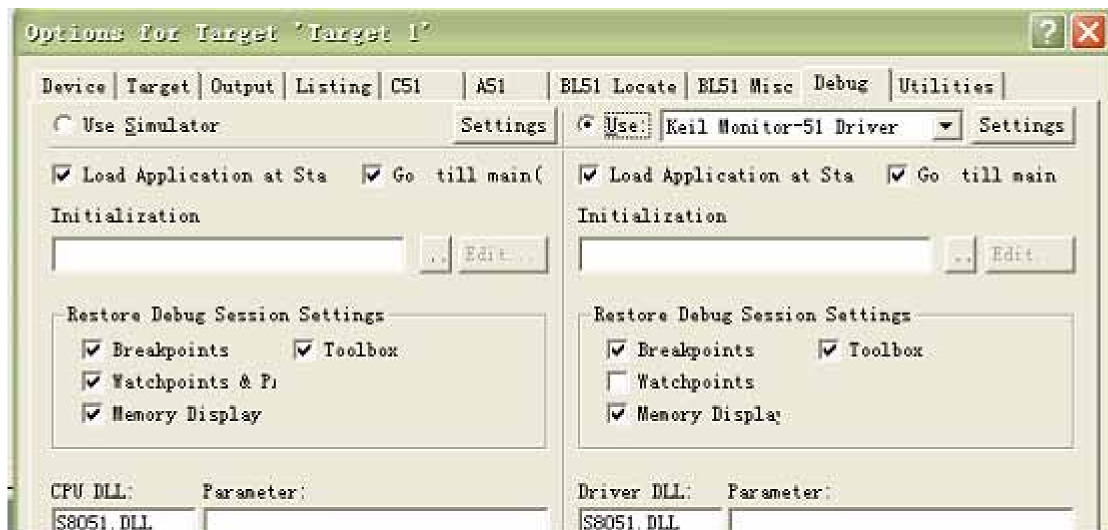


步骤二：配置

从 Keil 用户界面选择“Option for ‘Targe1t’”,

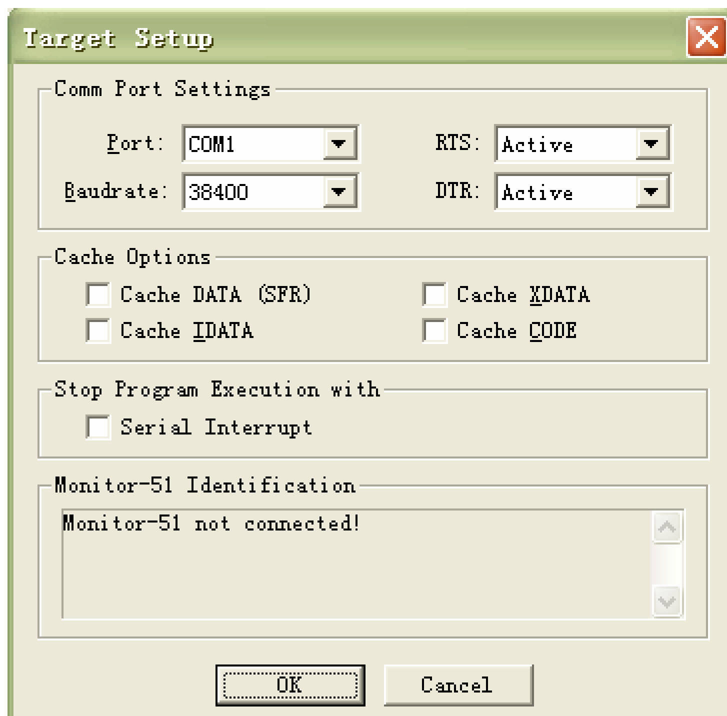


在“Option for ‘Targe1t ’”窗口 Debug 栏中，选择“Use Keil Monitor —51 Driver”，注意当每次 RESET 目标板时用户的代码区会被擦除，如果用户程序需要在开始时下载到目标板，要勾上“Load Application at Startup”。



点击 Setting 项，弹出 Target Setup 窗口进行配置，选择目标板与 PC 通讯的串行接口（COM1-COM4）和波特率，如果需要显示存储器的实时窗口，不要选择 Cache Option。注意如果选择了 Serial

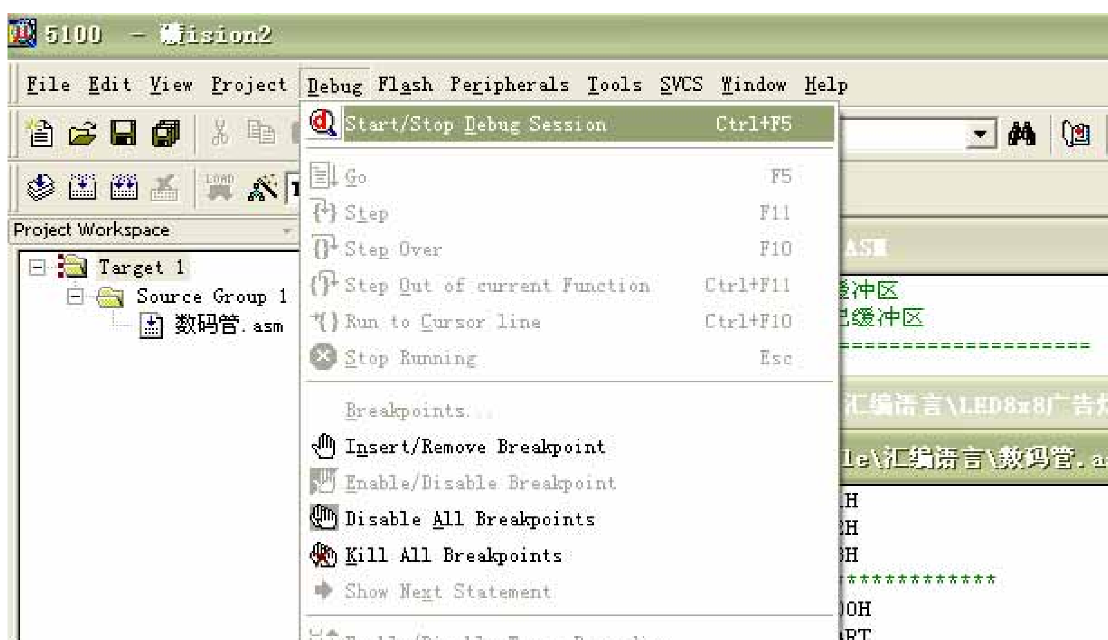
Interrupt 去 Stop Porgram Execution，SoftICE 将修改在位置 c：0023h 的中断向量 3 个 BYTE，请确保用户程序没有占用这些地址。



以上的设定可以在程序调试过程再次修改。

步骤 3：开始调试

从 Debug 菜单点击 Start/Stop Debug Session 开始调试过程。



在连接过程中，偶尔会出现连接重试的情况，此时只要按一下复位按钮即可连接成功，建议每次连接之前进行复位。连接成功之后就可直接单步、断点、全速运行。

特别提示：

不要对监控芯片进行烧写！

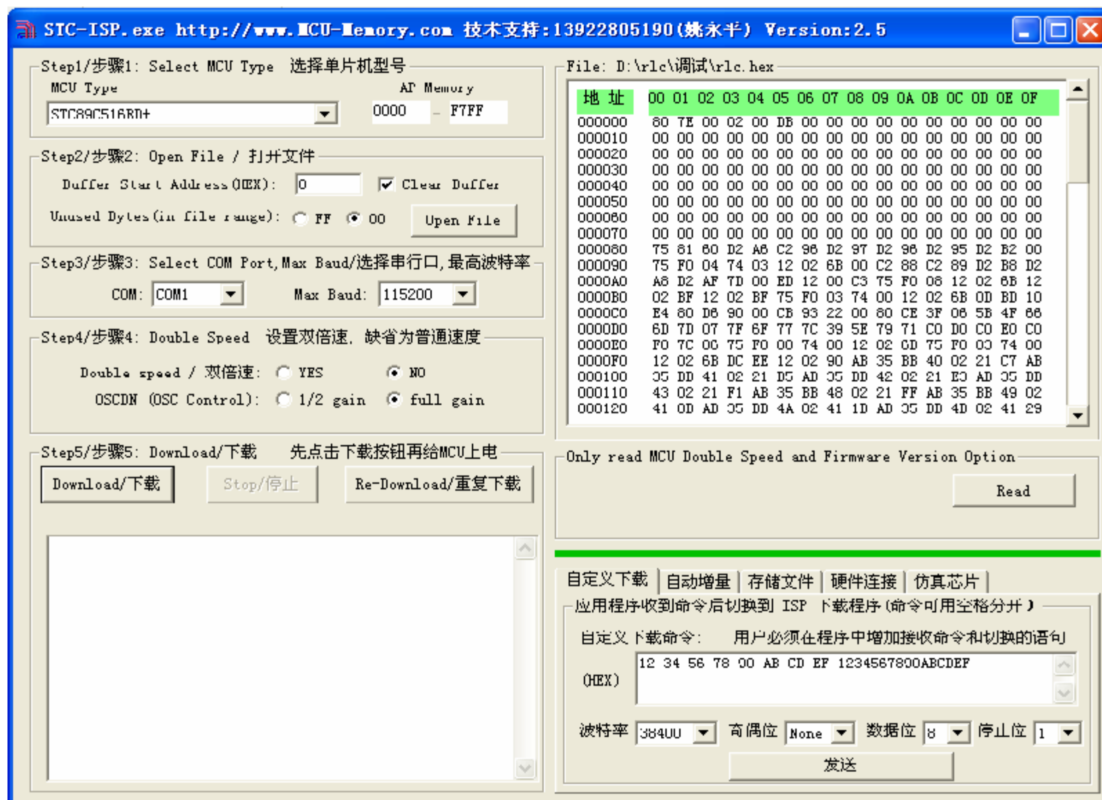
连接不成功的几种情况：

1. ***波特率设置不对；***
2. ***端口号设置不对；***
3. ***端口被其它应用程序占用；***
4. ***片内监控已被意外操作冲掉；***

第五章 在系统可编程的使用

STC89 系列单片机大部分具有在系统可编程（ISP）的特性，ISP 的好处是省去购买通用编程器，单片机在用户系统上即可下载/烧录用户程序。大部分 STC89 系列单片机在销售给用户之前已经在单片机内部固化有 ISP 系统引导程序，配合 PC 端的控制程序即可将用户的程序代码下载进单片机内部，故无须编程器（速度比通用编程器快）。不要用通用编程器编程，否则有可能将单片机内部已固化的 ISP 系统引导程序擦除，造成无法使用 STC 提供的 ISP 软件下载用户的程序代码。

操作步骤：解压、安装、运行 STC-ISP 软件



步骤 1：选择你所使用的单片机型号，STC89C52RC

步骤 2：打开文件，要烧录用户程序，必须调入用户的程序代码（*.bin,*.hex），可直接调光盘里可下载烧写的目标代码文件夹下的 hex 代码，进行测试

步骤 3：选择串口号，你所用的电脑串口是 COM1 还是 COM2

步骤 4：设置是否双倍速，双倍速选中 Double Speed 即可，STC89C52RC 系列出厂时为单倍速，用户可指定设为双倍速，如想从双倍速恢复成单倍速，则需用通用编程器擦除整个晶片方可，这会将单片机内部已烧录的 ISP 引导程序擦除。一般使用缺省设置即可，无须设置。

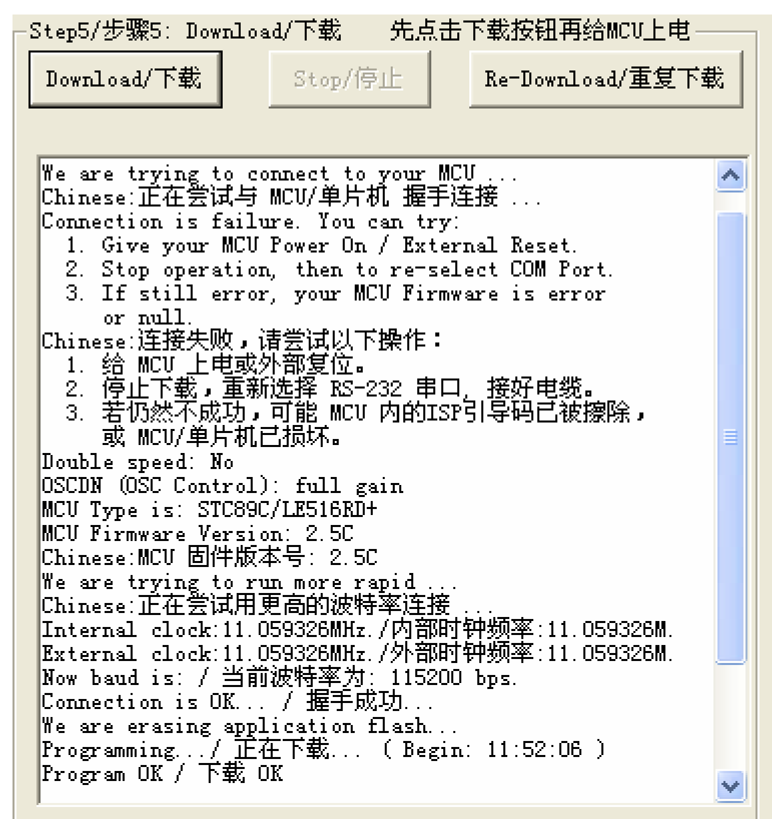
OSCDN：单片机时钟振荡器增益

选 1/2 gain 为降一半，降低 EMI；选 full gain（全增益）为正常状态

步骤 5：选择“ Download/下载 ”按钮下载用户的程序进单片机内部，可重复执行步骤 5，也可选择“ Re-Download/重复下载 ”按钮。

下载时注意看提示，主要看是否要给单片机上电或复位，下载速度比一般通用编程器快。一般先选择“ Download/下载 ”按钮，然后再给单片机上电复位（先彻底断电），而不要先上电。

下载编程成功如图所示：



这样即可上电运行用户的程序了。